Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Комп'ютерний практикум №2
З дисципліни «Теорія прийняття рішень»
на тему «Оптимізація за бінарним відношення»
1 варіант

Перевірила:

Жураковська О.С.

Виконав:

студент гр. ІС-01
Адамов Д.І.

Київ-2023

Завдання 1

Для кожного з бінарних відношень R1-R8 (із завдання 1 практикуму 1) визначити множину найкращих альтернатив за принципами домінування та блокування.

Відношення	Клас, до якого належить БВ	Опт. альтернативи за принципом домінування	Опт. Альтернативи за принципом блокування
R1	Квазіпорядок	$X^*_R = \{2, 3, 8\}$ $X^{**}_R = \emptyset$	$X^0_R = \{2, 3, 8\}$ $X^{00}_R = \emptyset$
R2	-	$X^*_R = \emptyset$ $X^{**}_R = \emptyset$	$X^0_R = \{6, 8\}$ $X^{00}_R = \emptyset$
R3	-	$X^*_R = \emptyset$ $X^{**}_R = \emptyset$	$X^0_R = \{1, 6\}$ $X^{00}_R = \emptyset$
R4	Нестрогий порядок	$X^*_R = \{3\}$ $X^{**}_R = \{3\}$	$X^0_R = \{3\}$ $X^{00}_R = \{3\}$
R5	Строгий порядок	$X^*_P = \{6\}$	$X^0_P = \{6\}$
R6	Нестрогий порядок	$X^*_R = \{8\}$ $X^{**}_R = \{8\}$	$X^0_R = \{8\}$ $X^{00}_R = \{8\}$
R7	Еквівалентність	$X^*_R = \emptyset$ $X^{**}_R = \emptyset$	$X^0_R = \{1, 2, 3, 4, 5, 6, 7, 8\}$ $X^{00}_R = \emptyset$
R8	Строгий порядок	$X^*_P = \emptyset$	$X^0_P = \{1, 5, 6\}$

Завдання 2 (варіанти завдань в файлі Практикум 2. Варіанти для завдання 2 Файл)

На множині із 15 альтернатив задано 10 бінарних відношень R_1-R_{10} матрицями відношень. Для кожного із БВ R_i необхідно:

1) перевірити наявність властивості ациклічності;
2)
   a) якщо відношення R_i є ациклічним - знайти множину Неймана-Моргенштерна (отриманий результат обгрунтувати – показати, що отримана множина відповідає означенню розв'язка Неймана-Моргенштерна);
   b) якщо відношення R_i не ациклічне - знайти множини оптимальних альтернатив за принципом К-оптимізації (k=1, k=2, k=3, k=4)

Відношення	Ациклічне/ неациклічне	Розв'язок Неймана-Моргенштерна	Опт. Альтернативи за принципом К-оптимізації
R1	+	$X^{НМ} = \{0, 8, 3, 11\}$	
R2	+	$X^{НМ} = \{0, 3\}$	
R3	+	$X^{НМ} = \{0, 3, 13\}$	
R4	+	$X^{НМ} = \{2, 4, 6, 8\}$	
R5	+	$X^{НМ} = \{4, 6, 12, 13\}$	
R6	-		1-max: {0, 2, 4, 11, 12, 13} 1-opt: {0, 2, 4, 11, 12, 13} 2-max: {0, 2, 4, 11, 12, 13} 2-opt: {0, 2, 4, 11, 12, 13} 3-max: {0, 6, 7, 11, 12} 3-opt: {} 4-max: {0, 11, 12} 4-opt: {}
R7	+	$X^{НМ} = \{0, 4, 8, 12\}$	
R8	-		1-max: {0, 1, 3, 6, 9, 13} 1-opt: {0, 1, 3, 6, 9, 13} 2-max: {0, 1, 3, 6, 9, 13} 2-opt: {0, 1, 3, 6, 9, 13} 3-max: {0, 1, 2, 11, 13} 3-opt: {} 4-max: {0, 1, 13} 4-opt: {}
R9	-		1-max: {2, 3, 6, 10, 11, 13} 1-opt: {2, 3, 6, 10, 11, 13} 2-max: {2, 6, 7, 13, 14} 2-opt: {} 3-max: {0, 1, 2, 11, 13} 3-opt: {} 4-max: {2, 13, 6} 4-opt: {}
R10	+	$X^{НМ} = \{1, 2, 6\}$	

Результати виконання завдання 1

1 Квазіпорядок

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

I			I	I	I		
P	I	I	P	P	P	P	I
P	I	I	P	P	P	P	I
I			I	I	I		
I			I	I	I		
I			I	I	I		
P			P	P	P	I	
P	I	I	P	P	P	P	I

Відношення не є асиметричним

Оптимізація за домінуванням:
X^{*}_{R} = {2, 3, 8} – у рядках всі елементи є одиницями
X^{**}_{R} = ∅, оскільки немає відповідних до X^{*}_{R} стовпчиків, де одиниця лише на головній діагоналі

Оптимізація за блокуванням:
X^{0}_{R} = {2, 3, 8} – стовпці містять лише I або 0
X^{00}_{R} = ∅ – немає стовпців, які містять усі 0, без врахування головної діагоналі

2 Не належить до класу

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

I	N		I		N	P	N
N	N	P	N	N	I		N
P		N	I	P	N	N	N
I	N	I	N	P	I		I
P	N			I	N	P	I
N	I	N	I	N	I	P	N
	P	N	P			I	N
N	N	N	I	I	N	N	I

Відношення не є асиметричним

Оптимізація за домінуванням:
X^{*}_{R} = ∅ – немає рядка, що містить усі одиниці
X^{**}_{R} = ∅, оскільки X^{*}_{R} = ∅

Оптимізація за блокуванням:
X^{0}_{R} = {6, 8} – стовпці не містять P
X^{00}_{R} = ∅ – немає стовпців, які містять усі 0, без врахування діагоналі

3 Не належить до класу

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

I	I	P	N	N	I	I	P
I	N	N	P	P	I	I	P
	N	N	N	N	N	I	P
N	P	N	N		N	P	N
N	P	N	P	I	N		N
I	I	N	N	N	I	N	I
I	I	I		P	N	I	
			N	N	I	P	I

Відношення не є асиметричним

Оптимізація за домінуванням:
$X^{*}_{R} = \emptyset$ – немає рядка, що містить усі одиниці
$X^{**}_{R} = \emptyset$, оскільки $X^{*}_{R} = \emptyset$

Оптимізація за блокуванням:
$X^{0}_{R} = \{1, 6\}$ – стовпці не містять P
$X^{00}_{R} = \emptyset$ – немає стовпців, які містять усі 0, без врахування головної діагоналі

4 Нестрогий порядок

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

I	P		P	P	P	P	P
	I		P	P	P		
P	P	I	P	P	P	P	P
			I				
			P	I			
			P	P	I		
	P		P	P	P	I	
	P		P	P	P	P	I

Відношення не є асиметричним

Оптимізація за домінуванням:
$X^{*}_{R} = \{3\}$ – у рядку всі одиниці
$X^{**}_{R} = \{3\}$ – у стовпці всі нулі, без врахування головної діагоналі

Оптимізація за блокуванням:
$X^{0}_{R} = \{3\}$ – у стовпці немає P
$X^{00}_{R} = \{3\}$ – у стовпці всі нулі, без врахування головної діагоналі

5 Строгий порядок

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

N	P	P	P				P
	N						
	P	N	P				P
	P		N				
P	P	P	P	N			P
P	P	P	P	P	N	P	P
P	P	P	P	P		N	P
	P		P				N

Відношення асиметричне

Оптимізація за домінуванням:
$X^*_P = \{6\}$ – у відповідному рядку всі одиниці, крім головної діагоналі

Оптимізація за блокуванням:
$X^0_P = \{6\}$ – у відповідному стовпці всі нулі

6 Нестрогий порядок

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

I	P	P	P	P	P	P	0
0	I	P	P	0	P	P	0
0	0	I	0	0	P	0	0
0	0	P	I	0	P	P	0
0	P	P	P	I	P	P	0
0	0	0	0	0	I	0	0
0	0	P	0	0	P	I	0
P	P	P	P	P	P	P	I

Відношення не є асиметричним

Оптимізація за домінуванням:
$X^*_R = \{8\}$ – рядок містить усі одиниці
$X^{**}_R = \{8\}$ – стовпець містить усі нулі, без врахування головної діагоналі

Оптимізація за блокуванням:
$X^0_R = \{8\}$ – стовпець не містить P
$X^{00}_R = \{8\}$ – стовпець містить усі нулі, без врахування головної діагоналі

7 Еквівалентність

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

I	0	0	0	0	0	I	0
0	I	I	I	0	0	0	I
0	I	I	I	0	0	0	I
0	I	I	I	0	0	0	I
0	0	0	0	I	I	0	0
0	0	0	0	I	I	0	0
I	0	0	0	0	0	I	0
0	I	I	I	0	0	0	I

Відношення не є асиметричним

Оптимізація за домінуванням:
$X^{*}_{R} = \emptyset$ – немає рядка, що містить усі одиниці
$X^{**}_{R} = \emptyset$, оскільки $X^{*}_{R} = \emptyset$

Оптимізація за блокуванням:
$X^{0}_{R} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ – стовпці не мають P
$X^{00}_{R} = \emptyset$ – немає стовпців, у яких всі нулі, без врахування головної діагоналі

8 Строгий порядок

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

0	P	P	P	0	0	P	P
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	P	P	P	0	0	P	P
0	P	P	P	0	0	P	P
0	P	P	P	0	0	0	0
0	P	P	P	0	0	0	0

Відношення є асиметричним

Оптимізація за домінуванням:
$X^{*}_{P} = \emptyset$ – немає рядка, що містить усі одиниці, крім головної діагоналі

Оптимізація за блокуванням:
$X^{0}_{P} = \{1, 5, 6\}$ – стовпці містять всі нулі

Результати виконання завдання 2

Relation #1

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

Relation #1 is acyclic. Neumann-Morgenstern optimization will be used.

1) Формуємо множину S0, в яку будуть входити елементи з порожнім верхнім перерізом (у матриці – у таких стовпчиках всі нулі).
2) Множина S1 будується на основі S0 – включаємо елементи, для яких верхній переріз включається в множину S0 (у матриці у стовпчиках одиниці стоять лише в тих рядках, що входять до множини S0).
3) Операція повторюється для Sn+1/Sn, допоки Sn не буде дорівнювати всій множині альтернатив Ω.

S0: {0}
S1: {0, 1}
S2: {0, 1, 3}
S3: {0, 1, 3, 4}
S4: {0, 1, 3, 4, 5}
S5: {0, 1, 3, 4, 5, 6}
S6: {0, 1, 3, 4, 5, 6, 8}
S7: {0, 1, 2, 3, 4, 5, 6, 8}
S8: {0, 1, 2, 3, 4, 5, 6, 7, 8}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 10}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}
Q1: {0}
Q2: {0, 3}
Q3: {0, 3}
Q4: {0, 3}
Q5: {0, 3}
Q6: {0, 8, 3}
Q7: {0, 8, 3}
Q8: {0, 8, 3}
Q9: {0, 8, 3}
Q10: {0, 8, 3, 11}
Q11: {0, 8, 3, 11}
Q12: {0, 8, 3, 11}
Q13: {0, 8, 3, 11}

Neumann-Morgenstern solution: {0, 8, 3, 11}
Internal stability: True
External stability: True

Relation #2
0 1 1 0 1 1 1 0 0 0 1 1 1 0 1
0 0 1 0 1 0 0 0 1 1 1 0 0 0 0
0 0 0 1 0 1 1 0 1 0 0 1 1 0 1
0 0 0 0 0 1 1 1 1 1 0 1 1 1 0
0 0 1 0 0 1 1 0 0 0 1 1 0 1 1
0 0 0 0 0 0 0 0 1 1 1 0 0 0 1
0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0
0 0 0 0 0 1 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1
0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 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 1 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #2 is acyclic. Neumann-Morgenstern optimization will be used.
S0: {0}
S1: {0, 1}

S2: {0, 1, 4}
S3: {0, 1, 2, 4}
S4: {0, 1, 2, 3, 4}
S5: {0, 1, 2, 3, 4, 6}
S6: {0, 1, 2, 3, 4, 6, 7}
S7: {0, 1, 2, 3, 4, 5, 6, 7}
S8: {0, 1, 2, 3, 4, 5, 6, 7, 9}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 13}
S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13}
S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13}
S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
S14: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}
Q1: {0}
Q2: {0}
Q3: {0}
Q4: {0, 3}
Q5: {0, 3}
Q6: {0, 3}
Q7: {0, 3}
Q8: {0, 3}
Q9: {0, 3}
Q10: {0, 3}
Q11: {0, 3}
Q12: {0, 3}
Q13: {0, 3}
Q14: {0, 3}

Neumann-Morgenstern solution: {0, 3}
Internal stability: True
External stability: True

Relation #3

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

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

Relation #3 is acyclic. Neumann-Morgenstern optimization will be used.
S0: {0}
S1: {0, 1}
S2: {0, 1, 2}
S3: {0, 1, 2, 5}
S4: {0, 1, 2, 3, 5}
S5: {0, 1, 2, 3, 4, 5}
S6: {0, 1, 2, 3, 4, 5, 6}
S7: {0, 1, 2, 3, 4, 5, 6, 7}
S8: {0, 1, 2, 3, 4, 5, 6, 7, 9}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12}
S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12}
S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
S14: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}
Q1: {0}
Q2: {0}
Q3: {0}
Q4: {0, 3}
Q5: {0, 3}
Q6: {0, 3}
Q7: {0, 3}
Q8: {0, 3}
Q9: {0, 3}
Q10: {0, 3}
Q11: {0, 3}
Q12: {0, 3}
Q13: {0, 3, 13}
Q14: {0, 3, 13}

Neumann-Morgenstern solution: {0, 3, 13}
Internal stability: True

External stability: True

Relation #4

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

Relation #4 is acyclic. Neumann-Morgenstern optimization will be used.
S0: {2, 4}
S1: {1, 2, 4}
S2: {1, 2, 3, 4, 5}
S3: {1, 2, 3, 4, 5, 6}
S4: {0, 1, 2, 3, 4, 5, 6}
S5: {0, 1, 2, 3, 4, 5, 6, 7}
S6: {0, 1, 2, 3, 4, 5, 6, 7, 9}
S7: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
S8: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14}
S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {2, 4}
Q1: {2, 4}
Q2: {2, 4}
Q3: {2, 4, 6}
Q4: {2, 4, 6}
Q5: {2, 4, 6}
Q6: {2, 4, 6}
Q7: {8, 2, 4, 6}
Q8: {8, 2, 4, 6}
Q9: {8, 2, 4, 6}

Q10: {8, 2, 4, 6}
Q11: {8, 2, 4, 6}

Neumann-Morgenstern solution: {8, 2, 4, 6}
Internal stability: True
External stability: True

Relation #5
0 0 1 0 0 1 1 1 0 1 0 1 1 0 0
0 0 1 1 0 1 0 0 0 1 1 1 0 0 0
0 0 0 1 0 0 1 1 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
1 1 1 1 0 1 0 1 1 1 1 1 0 0 1
0 0 0 1 0 0 0 1 1 0 0 1 0 0 0
0 0 0 1 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 1 0 0 1 1 1 1
1 0 0 1 0 0 0 1 1 1 0 0 1 1 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 1 0 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

Relation #5 is acyclic. Neumann-Morgenstern optimization will be used.
S0: {4}
S1: {1, 4}
S2: {1, 4, 10}
S3: {0, 1, 4, 10}
S4: {0, 1, 2, 4, 5, 10}
S5: {0, 1, 2, 4, 5, 6, 10}
S6: {0, 1, 2, 3, 4, 5, 6, 7, 10}
S7: {0, 1, 2, 3, 4, 5, 6, 7, 9, 10}
S8: {0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 13}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14}
S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {4}
Q1: {4}
Q2: {4}
Q3: {4}
Q4: {4}

Q5: {4, 6}
Q6: {4, 6}
Q7: {4, 6}
Q8: {4, 13, 6}
Q9: {4, 13, 6}
Q10: {4, 13, 6}
Q11: {12, 4, 13, 6}

Neumann-Morgenstern solution: {12, 4, 13, 6}
Internal stability: True
External stability: True

Relation #6
0 1 0 1 0 1 1 1 1 1 1 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 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 1 1 1 1 1 0 0 0 1
0 1 0 1 0 1 1 1 1 1 1 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 1 0 1 0 1 1 1 1 1 1 0 0 0 1
0 1 0 1 0 1 1 1 1 1 1 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

Relation #6 has a cycle. k-optimization will be used.
PIN matrix:
N P N P N P P P P P P N N N P
N N N N N N N N N N N
N N N N N N N N N N N N N N N
N N N N N N N N N N N
N N N N N N N N N N N N N N N
N N N N N N N N N N N
P N P N P I I P P P N P
P N P N P I I P P P N P
N N N N N N N N N N N
N N N N N N N N N N N
N N N N N N N N N N N
N P N P N P P P P P P N N N P

N P N P N P P P P P P N N N P
N N N N N N N N N N N N N N N
N N N N N N N N N

k1 (R (P, I, N)).
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
0 1 1 1 1 1 1 1 1 1 1 0 0 1 1
0 1 1 1 1 1 1 1 1 1 1 0 0 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

Матриця для k1 будується таким чином: 1, якщо елемент це P, I або N, інакше - 0.
Сім’я множин $S^1_R(x)$ представлена як рядки матриці.
Щоб обрати найкращу альтернативу, треба обрати множину, що є максимальною за включенням, тобто такий рядок включає в себе всі інші рядки і при цьому жоден інший рядок не включає його як власну підмножину. Таких альтернатив може бути декілька.

1-оптимальні альтернативи – це такі альтернативи, $S^1_R(x)$ яких співпадає з усією множиною альтернатив Ω. Їх може не існувати.

k1 max elements: {0, 2, 4, 11, 12, 13}
k1 opt elements: {0, 2, 4, 11, 12, 13}

k2 (R (P, N)).
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1
0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

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

k2 max elements: {0, 2, 4, 11, 12, 13}
k2 opt elements: {0, 2, 4, 11, 12, 13}

k3 (R (P, I))
0 1 0 1 0 1 1 1 1 1 1 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 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 1 1 1 1 1 0 0 0 1
0 1 0 1 0 1 1 1 1 1 1 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 1 0 1 0 1 1 1 1 1 1 0 0 0 1
0 1 0 1 0 1 1 1 1 1 1 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

k3 max elements: {0, 6, 7, 11, 12}
k3 opt elements: {}

k4 (R (P))
0 1 0 1 0 1 1 1 1 1 1 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 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 0 0 1 1 1 0 0 0 1
0 1 0 1 0 1 0 0 1 1 1 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 1 0 1 0 1 1 1 1 1 1 0 0 0 1
0 1 0 1 0 1 1 1 1 1 1 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

k4 max elements: {0, 11, 12}
k4 opt elements: {}

Relation #7
0 1 1 1 0 1 1 0 0 1 1 0 0 0 1
0 0 1 1 1 1 0 0 0 1 0 0 1 1 1
0 0 0 1 0 1 1 0 0 0 0 1 0 1 0
0 0 0 0 0 0 1 1 1 0 1 0 0 0 0
0 0 1 1 0 1 1 1 0 1 0 0 0 0 0
0 0 0 1 0 0 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 1 0 0 1
0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #7 is acyclic. Neumann-Morgenstern optimization will be used.
S0: {0}
S1: {0, 1}
S2: {0, 1, 4}
S3: {0, 1, 2, 4}
S4: {0, 1, 2, 4, 5}
S5: {0, 1, 2, 3, 4, 5}
S6: {0, 1, 2, 3, 4, 5, 8}
S7: {0, 1, 2, 3, 4, 5, 8, 13}
S8: {0, 1, 2, 3, 4, 5, 6, 8, 13}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 13}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 13}
S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13}
S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13}
S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}

Q1: {0}
Q2: {0, 4}
Q3: {0, 4}
Q4: {0, 4}
Q5: {0, 4}
Q6: {0, 8, 4}
Q7: {0, 8, 4}
Q8: {0, 8, 4}
Q9: {0, 8, 4}
Q10: {0, 8, 4}
Q11: {0, 8, 4}
Q12: {0, 8, 4}
Q13: {0, 8, 4, 12}

Neumann-Morgenstern solution: {0, 8, 4, 12}
Internal stability: True
External stability: True

Relation #8
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 1 0 1 1 0 1 1 0 1 1 1 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 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 0 0 0 0
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #8 has a cycle. k-optimization will be used.
PIN matrix:
N N P N P P N P P N P P P N P
N N P N P P N P P N P P P N P
I N P P N P P N P I P P
N N N N N N N N N N N N N N N
N N N N N N N N N N N
N N N N N N N N N N N

N N N N N N N N N N N N N N N
N N N N N N N N N N N
N N N N N N N N N N N
N N N N N N N N N N N N N N N
N N N N N N N N N N N
I N P P N P P N P I P P
N N N N N N N N N N N
N N P N P P N P P N P P P N P
N N N N N N N N N N N

k1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
0 0 1 1 1 1 1 1 1 1 1 1 1 0 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

k1 max elements: {0, 1, 3, 6, 9, 13}
k1 opt elements: {0, 1, 3, 6, 9, 13}

k2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1
0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

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

k2 max elements: {0, 1, 3, 6, 9, 13}
k2 opt elements: {0, 1, 3, 6, 9, 13}

k3
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 1 0 1 1 0 1 1 0 1 1 1 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 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 1 1 0 1 1 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

k3 max elements: {0, 1, 2, 11, 13}
k3 opt elements: {}

k4
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 0 0 1 1 0 1 1 0 1 0 1 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 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 1 0 1 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 1 0 1 1 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

k4 max elements: {0, 1, 13}
k4 opt elements: {}

Relation #9
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 1 0 0 1 1 0 1 1 1 0 0 1 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
1 1 0 0 1 1 0 1 1 1 0 0 1 0 1
1 1 0 0 1 1 0 1 1 1 0 0 1 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 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 1 0 0 1 1 0 1 1 1 0 0 1 0 1
1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

Relation #9 has a cycle. k-optimization will be used.
PIN matrix:
N N N N N N N N N N
N N N N N N N N N N
P P N N P P N P P P N N P N P
N N N N N N N N N N N N N N N
N N N N N N N N N N
N N N N N N N N N N
P P N N P P N P P P N N P N P
P P N P P I P P N N P I
N N N N N N N N N N
N N N N N N N N N N
N N N N N N N N N N N N N N N
N N N N N N N N N N N N N N N
N N N N N N N N N N
P P N N P P N P P P N N P N P
P P N P P I P P N N P I

k1
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

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

k1 max elements: {2, 3, 6, 10, 11, 13}
k1 opt elements: {2, 3, 6, 10, 11, 13}

k2
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

k2 max elements: {2, 3, 6, 10, 11, 13}
k2 opt elements: {2, 3, 6, 10, 11, 13}

k3
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 1 0 0 1 1 0 1 1 1 0 0 1 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

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

k3 max elements: {2, 6, 7, 13, 14}
k3 opt elements: {}

k4
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 1 0 0 1 1 0 1 1 1 0 0 1 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
1 1 0 0 1 1 0 1 1 1 0 0 1 0 1
1 1 0 0 1 1 0 0 1 1 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 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 1 1 0 1 1 1 0 0 1 0 1
1 1 0 0 1 1 0 0 1 1 0 0 1 0 0

k4 max elements: {2, 13, 6}
k4 opt elements: {}

Relation #10
0 0 0 1 1 0 1 0 1 1 1 0 0 1 1
1 0 0 0 1 0 0 0 1 0 0 0 1 1 1
1 0 0 1 0 1 0 1 0 0 1 1 1 0 0
0 0 0 0 0 1 0 1 0 0 1 1 1 0 1
0 0 0 0 0 1 1 1 1 1 1 0 1 0 0
0 0 0 0 0 0 1 1 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 1 1 1 0 1 0 1
0 0 0 0 0 0 0 0 1 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0

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

Relation #10 is acyclic. Neumann-Morgenstern optimization will be used.
S0: {1, 2}
S1: {0, 1, 2}
S2: {0, 1, 2, 3, 4}
S3: {0, 1, 2, 3, 4, 5}
S4: {0, 1, 2, 3, 4, 5, 6, 7}
S5: {0, 1, 2, 3, 4, 5, 6, 7, 8}
S6: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
S7: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
S8: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14}
S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {1, 2}
Q1: {1, 2}
Q2: {1, 2}
Q3: {1, 2}
Q4: {1, 2, 6}
Q5: {1, 2, 6}
Q6: {1, 2, 6}
Q7: {1, 2, 6}
Q8: {1, 2, 6}
Q9: {1, 2, 6}
Q10: {1, 2, 6}
Q11: {1, 2, 6}

Neumann-Morgenstern solution: {1, 2, 6}
Internal stability: True
External stability: True

Програмний код

```
import numpy as np
from typing import List, Set


def read_relations(filename, size, count):
    relations = [np.zeros((size, size)) for _ in range(count)]
```

```
    def read_one_relation(file):
        # skip first row of a relation because it only contains name and dashes (---)
        file.readline()
```

```
        for row in range(size):
            line = file.readline()
            column = 0
            for char in line:
                if char != ' ' and (char == '1' or char == '0'):
                    relations[num_rel][row][column] = int(char)
                    column += 1
```

```
    with open(filename, "r") as file:
        for num_rel in range(count):
            read_one_relation(file)
```

```
    return relations
```

```
def print_relation(relation, num_rel=0):
    if num_rel != 0:
        print_to_file(f"Relation #{num_rel}")
    for row in relation:
        for element in row:
            if isinstance(element, int) or isinstance(element, float):
                print_to_file("{:.0f}".format(element) + " ", end="")
            else:
                print_to_file(element + " ", end="")
        print_to_file()
    print_to_file()
```

```
def print_to_file(data="", end="\n"):
    with open("results.txt", "a") as file:
        file.write(str(data))
        file.write(end)
# ----------
```

```
def has_cycle(relation):
```

```
    def does_vertex_has_cycle(vertex_index):
        reachable_vertexes = set()
        visited_vertexes = set()
```

```
        for (index, value) in enumerate(relation[vertex_index]):
            if value == 1:
                reachable_vertexes.add(index)
```

```
        visited_vertexes.add(vertex_index)
```

```
        if reachable_vertexes.__contains__(vertex_index):
            return True

        while len(visited_vertexes) != len(relation) and len(reachable_vertexes) > 0:
            unvisited_reachable_vertexes = reachable_vertexes.difference(
                visited_vertexes)
            if len(unvisited_reachable_vertexes) == 0:
                return False
            current_vertex = unvisited_reachable_vertexes.pop()
            visited_vertexes.add(current_vertex)

            for (index, value) in enumerate(relation[current_vertex]):
                if value == 1:
                    if index == vertex_index:
                        return True
                    reachable_vertexes.add(index)

            visited_vertexes.add(current_vertex)

        return False

    for vertex_index in range(len(relation)):
        if does_vertex_has_cycle(vertex_index):
            return True
    return False


def solve_NM(relation):

    def get_upper_cut(vertex):
        return set(index for (index, value) in enumerate(relation[:, vertex]) if value == 1)

    def get_S_sets():
        all_S_sets = [set()]

        # S[0]
        for vertex in range(len(relation)):
            if len(get_upper_cut(vertex)) == 0:
                all_S_sets[0].add(vertex)

        # S[k]
        while len(all_S_sets[-1]) != len(relation):
            vertexes_dominated_only_by_prev_S = set(vertex for vertex in range(len(relation))
                                                    if
get_upper_cut(vertex).issubset(all_S_sets[-1]))

            all_S_sets.append(
                all_S_sets[-1].union(vertexes_dominated_only_by_prev_S))

        return all_S_sets

    def get_Q_sets(all_S_sets: List[Set]):
        all_Q_sets = [all_S_sets[0]]  # need to make a copy explicitly?
        for i in range(1, len(all_S_sets)):
            diff_S_curr_and_S_prev = all_S_sets[i].difference(all_S_sets[i-1])

            just_the_required_condition = set(vertex for vertex in diff_S_curr_and_S_prev
                                              if
len(get_upper_cut(vertex).intersection(all_Q_sets[-1])) == 0)

            all_Q_sets.append(
```

```
                all_Q_sets[-1].union(just_the_required_condition))
        return all_Q_sets
```

```
    def check_internal_stability(vertexes):
        for row in vertexes:
            for column in vertexes:
                if relation[row][column] == 1:
                    return False
        return True
```

```
    def check_external_stability(vertexes):
        other_vertexes = set(range(len(relation))).difference(vertexes)
        for column in other_vertexes:
            found_better_in_set = False
            for row in vertexes:
                if relation[row][column] == 1:
                    found_better_in_set = True
                    break
            if found_better_in_set == False:
                return False
        return True
```

```
    def print_sets(sets, type):
        for (index, set) in enumerate(sets):
            print_to_file(f"{type}{index}: {set}")
        print_to_file()
```

```
    S_sets = get_S_sets()
    print_sets(S_sets, "S")
```

```
    Q_sets = get_Q_sets(S_sets)
    solution = Q_sets[-1]
    print_sets(Q_sets, "Q")
```

```
    int_stab = check_internal_stability(solution)
    ext_stab = check_external_stability(solution)
```

```
    print_to_file(f"Neumann-Morgenstern solution: {solution}")
    print_to_file(f"Internal stability: {int_stab}")
    print_to_file(f"External stability: {ext_stab}")
    print_to_file()
```

```
def solve_k_optimization(relation):
```

```
    def get_PIN_matrix():
        PIN_matrix = [[] for _ in range(len(relation))]
```

```
        for row in range(len(relation)):
            for column in range(len(relation)):
                if relation[row][column] == 1 and relation[column][row] == 0:
                    PIN_matrix[row].append("P")
                elif relation[row][column] == 1 and relation[column][row] == 1:
                    PIN_matrix[row].append("I")
                elif relation[row][column] == 0 and relation[column][row] == 0:
                    PIN_matrix[row].append("N")
                else:
                    PIN_matrix[row].append(" ")
        return PIN_matrix
```

```
    def get_H_matrix(k):
```

```
        H_matrix = []
        PIN_matrix = get_PIN_matrix()

        for row in range(len(relation)):
            H_matrix.append([0 for _ in range(len(relation))])
            for column in range(len(relation)):
                if k == 1 and (PIN_matrix[row][column] == "P" or PIN_matrix[row][column] == "I"
or PIN_matrix[row][column] == "N"):
                    H_matrix[row][column] = 1
                elif k == 2 and (PIN_matrix[row][column] == "P" or PIN_matrix[row][column] ==
"N"):
                    H_matrix[row][column] = 1
                elif k == 3 and (PIN_matrix[row][column] == "P" or PIN_matrix[row][column] ==
"I"):
                    H_matrix[row][column] = 1
                elif k == 4 and (PIN_matrix[row][column] == "P"):
                    H_matrix[row][column] = 1
                else:
                    H_matrix[row][column] = 0
        return H_matrix

    def is_subset_or_equal(list1, list2):
        for i in range(len(list1)):
            if list1[i] == 1 and list2[i] == 0:
                return False
        return True

    def get_max_and_opt(k):
        max_alts = set()
        max_alt_size = 0
        opt_alts = set()

        H_matrix = get_H_matrix(k)

        for vertex in range(len(relation)):

            if H_matrix[vertex].count(1) > 0:
                if len(max_alts) == 0:
                    max_alts.add(vertex)
                    max_alt_size = H_matrix[vertex].count(1)

                elif H_matrix[vertex].count(1) > max_alt_size:
                    max_alts = set([vertex])
                    max_alt_size = H_matrix[vertex].count(1)

                elif H_matrix[vertex].count(1) == max_alt_size:

                    if is_subset_or_equal(H_matrix[vertex],  H_matrix[list(max_alts)[0]]):
                        max_alts.add(vertex)

        if max_alt_size > 0:
            for H_row in H_matrix:
                if not is_subset_or_equal(H_row, H_matrix[list(max_alts)[0]]):
                    max_alts = set()
                    max_alt_size = 0
                    break

        if max_alt_size == len(relation):
            opt_alts = max_alts

        return max_alts, opt_alts
```

```
    print_to_file("PIN matrix:")
    print_relation(get_PIN_matrix())
```

```
    for i in range(1, 5):
        print_to_file(f"k{i}")
        print_relation(get_H_matrix(i))
        max_alts, opt_alts = get_max_and_opt(i)
        print_to_file(f"k{i} max elements: {max_alts}")
        print_to_file(f"k{i} opt elements: {opt_alts}")
        print_to_file()
```

```
def main():
    size = 15
    count = 10
```

```
    with open("results.txt", "w") as file:
        file.truncate()
```

```
    relations = read_relations("Варіант №1.txt", size, count)
```

```
    for (index, relation) in enumerate(relations):
```

```
        print_relation(relation, index+1)
        if has_cycle(relation):
            print_to_file(
                f"Relation #{index+1} has a cycle. k-optimization will be used.")
            solve_k_optimization(relation)
        else:
            print_to_file(
                f"Relation #{index+1} is acyclic. Neumann-Morgenstern optimization will be
used.")
            solve_NM(relation)
```

```
# --------------------------------
main()
```