Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра системного проектування

**Лабораторна робота №1**

з дисципліни “Теорія прийняття рішень”

на тему: “**Багатокритеріальний вибір. Визначення оптимальних альтернатив за Парето та Слейтером**”

Виконала:

студентка IV курсу, групи ДА-02

Тищенко Н.А.

Прийняла(в):

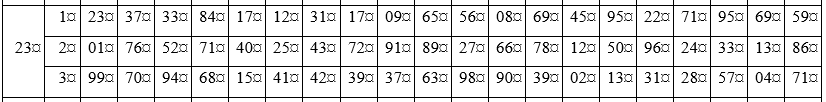
Київ-2023

Мета роботи

Ознайомитись з поняттями оптимальності за Парето та за Слейтером при багатокритеріальному виборі.

Завдання (варіант 23)

1. Для кожного рядка (1-3) за варіантом («Додаток А») побудувати таблицю значень альтернатив (A1-A20) в області критеріїв (Q1, Q2), де значення за першим критерієм відповідають першій цифрі числа, за другим критерієм – другій цифрі числа. Аналітично (за допомогою алгоритмів п. 3.1–3.2) та графічно визначити множину оптимальних рішень за Парето та за Слейтером (6 рисунків).
2. Для рядка, що складається з рядків 1-3 («Додаток А») за варіантом, побудувати таблицю значень альтернатив (A1-A60) в області критеріїв (Q1, Q2), де значення за першим критерієм відповідають першій цифрі числа, за другим критерієм – другій цифрі числа. Аналітично (за допомогою алгоритмів п. 3.1–3.2) та графічно визначити множину оптимальних рішень за Парето та за Слейтером (2 рисунки).



# **Контрольні запитання**

1. *З чого складається постановка задачі багатокритеріального вибору?*

Постановка задачі багатокритеріального складається з наступних трьох пунктів:

1) множину можливих рішень Х;

2) векторний критерій f;

3) відношення переваги .

1. *В чому полягає роль ОПР (особа, що приймає рішення)?*

Особі, що приймає рішення, з наявної множини можливих рішень потрібно обрати одне або декілька оптимальних. ОПТ зацікавлена в отриманні максимальних значень всіх компонентів векторного критерію *f.*

Сформована через аксіому вимога до поведінки ОПР:

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

1. *Що таке «оптимальність» при багатокритеріальному виборі?*

«Оптимальність» за Парето – не існує можливого рішення , для якого має місце нерівність  (парето-оптимальне рішення). Існує також «слабкий» варіант парето-оптимального рішення – за Слейтером - не існує такого можливого вирішення , для якого має місце нерівність .

1. *Що таке множина оптимальних рішень за Парето та за Слейтером?*

Усі парето-оптимальні або парето-ефективні рішення утворюють множину Парето , всі оптимальні за Слейтером рішення утворюють множину Слейтера .

1. *Як співвідносяться множини допустимих, оптимальних за Парето, оптимальних за Слейтером та обираємих рішень?*

Цілком зрозуміло, що множина усіх допустимих рішень буде найбільшою, оскільки вона не має абсолютно ніяких критеріїв вибору (відносно Парето та Слейтера). Множина оптимальних за Слейтером рішень буде дещо меншою за множину усіх допустимих рішень, оскільки містить у собі встановлене жорстке обмеження, а множина оптимальних за Паретом буде ще меншою за множину Слейтера, у той час як множина реально обираємих рішень буде найменшою з усіх представлених, оскільки ОПР має обрати серед найкращих рішень встановлену завданням кількість.



Рисунок 1 - Відношення між множинами допустимих, оптимальних за Слейтером, парето-оптимальних та обираємих рішень

1. *Що таке «слабке» парето-оптимальне рішення?*

Так називають оптимальне за Слейтером рішення, оскільки Слейтер (внаслідок більш жорсткого обмеження) пропонує менш цікаві рішення за ті, що може запропонувати Парето, але, як зазначалось, в багатьох випадках при вирішенні задач багатокритеріальної оптимізації використовуються саме оптимальні за Слейтером множини рішень.

1. *В чому полягає принцип Еджворта-Парето?*

Принцип Еджворта-Парето полягає у необхідності при вирішенні багатокритеріальних завдань виконувати оптимальний вибір в рамках множини Парето та можливих її підмножинах у вигляді множини Слейтера. Для нагадування, парето-оптимальне рішення – таке рішення, яке не може бути покращене (збільшене) по жодному з наявних критеріїв без погіршення (зменшення) по будь-якому принаймні одному іншому критерію.

Хід роботи

1. Побудуємо значення альтернатив в області критеріїв Q1, Q2, де значення за першим критерієм відповідають першій цифрі числа, за другим критерієм – другій цифрі числа.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ряд 1 | | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 | A17 | A18 | A19 | A20 |
| Критерії | q1 | 2 | 3 | 3 | 8 | 1 | 1 | 3 | 1 | 0 | 6 | 5 | 0 | 6 | 4 | 9 | 2 | 7 | 9 | 6 | 5 |
| q2 | 3 | 7 | 3 | 4 | 7 | 2 | 1 | 7 | 9 | 5 | 6 | 8 | 9 | 5 | 5 | 2 | 1 | 5 | 9 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ряд 2 | | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 | A17 | A18 | A19 | A20 |
| Критерії | q1 | 0 | 7 | 5 | 7 | 4 | 2 | 4 | 7 | 9 | 8 | 2 | 6 | 7 | 1 | 5 | 9 | 2 | 3 | 1 | 8 |
| q2 | 1 | 6 | 2 | 1 | 0 | 5 | 3 | 2 | 1 | 9 | 7 | 6 | 8 | 2 | 0 | 6 | 4 | 3 | 3 | 6 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ряд 3 | | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 | A17 | A18 | A19 | A20 |
| Критерії | q1 | 9 | 7 | 9 | 6 | 1 | 4 | 4 | 3 | 3 | 6 | 9 | 9 | 3 | 0 | 1 | 3 | 2 | 5 | 0 | 7 |
| q2 | 9 | 0 | 4 | 8 | 5 | 1 | 2 | 9 | 7 | 3 | 8 | 0 | 9 | 2 | 3 | 1 | 8 | 7 | 4 | 1 |

Усі ряди разом:

Зображення, що містить текст, число, знімок екрана, Шрифт

Автоматично згенерований опис

1. Визначимо аналітично множину оптимальних рішень та побудуємо відповідні таблиці

Зображення, що містить текст, знімок екрана, число, ряд

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, ряд, число

Автоматично згенерований опис

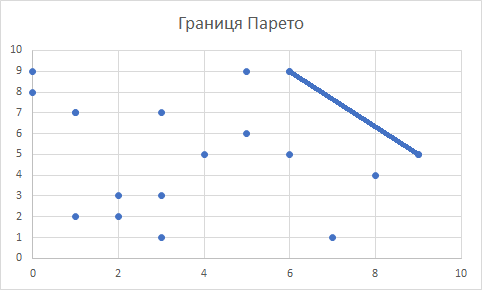
Зображення, що містить текст, знімок екрана, ряд, число

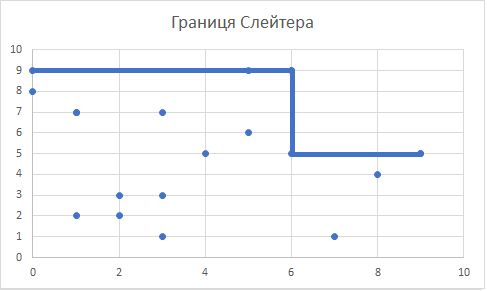
Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, число, Паралель

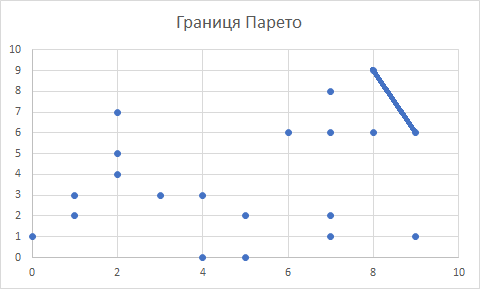
Автоматично згенерований опис

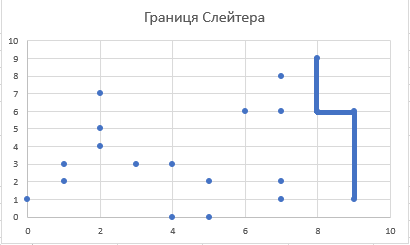
3.



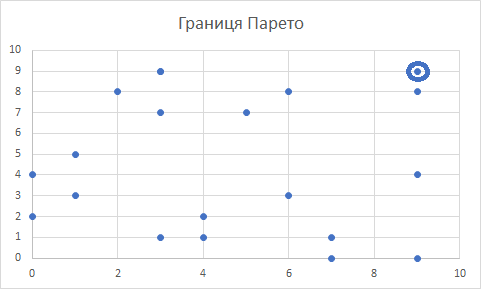


Ряд 2





Ряд 3



Зображення, що містить текст, знімок екрана, число, ряд

Автоматично згенерований опис

Усі ряди





Код

const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

const transformMatrixByCriteria = function (matrix, rowNumber){

let result = {

q1: [],

q2: []

};

let splitedNumbers = []

if(rowNumber === '\*'){

matrix.map(row => {

row.map(number => {

splitedNumbers.push(number.toString().split(''));

}

)

})

}else {

matrix[Number(rowNumber) - 1].map(number => {

splitedNumbers.push(number.toString().split(''));

})

}

let i = 0;

while (i < splitedNumbers.length){

if(splitedNumbers[i].length === 1){

result.q1.push(0);

result.q2.push(Number(splitedNumbers[i][0]))

}else {

result.q1.push(Number(splitedNumbers[i][0]));

result.q2.push(Number(splitedNumbers[i][1]))

}

i++;

}

return result;

}

const selectBestAlternatives = (transformedMatrix, row, compareIwithJ, compareJwithI) => {

let i = 0;

let j = 1;

let result = [...row];

while(i < row.length - 1){

if (j < row.length){

if(compareIwithJ(i, j)){

result[j] = result[j].toString();

j++;

}else if(compareJwithI(i, j)){

result[i] = result[i].toString();

i++;

j = i + 1;

}else{

j++;

}

}else {

i = i + 1;

j = i + 1;

}

}

console.log('result: ', result)

return result;

}

const pareto = function (transformedMatrix, row){

const compareIwithJ = (i, j) => transformedMatrix.q1[i] >= transformedMatrix.q1[j] && transformedMatrix.q2[i] >= transformedMatrix.q2[j];

const compareJwithI = (i, j) => transformedMatrix.q1[j] >= transformedMatrix.q1[i] && transformedMatrix.q2[j] >= transformedMatrix.q2[i];

const result = selectBestAlternatives(transformedMatrix, row, compareIwithJ, compareJwithI);

let returnedString = 'Pareto: '

result.map(item => {

if(typeof item === 'number'){

returnedString = returnedString + item + ', ';

}

});

console.log(returnedString)

};

const slater = function (transformedMatrix, row){

const compareIwithJ = (i, j) => transformedMatrix.q1[i] > transformedMatrix.q1[j] && transformedMatrix.q2[i] > transformedMatrix.q2[j];

const compareJwithI = (i, j) => transformedMatrix.q1[j] > transformedMatrix.q1[i] && transformedMatrix.q2[j] > transformedMatrix.q2[i];

const result = selectBestAlternatives(transformedMatrix, row, compareIwithJ, compareJwithI);

let returnedString = 'Slater: '

result.map(item => {

if(typeof item === 'number'){

returnedString = returnedString + item + ', ';

}

});

console.log(returnedString)

}

function main () {

const numbers = [

[23, 37, 33, 84, 17, 12, 31, 17, 9, 65, 56, 8, 69, 45, 95, 22, 71, 95, 69, 59],

[1, 76, 52, 71, 40, 25, 43, 72, 91, 89, 27, 66, 78, 12, 50, 96, 24, 33, 13, 86],

[99, 70, 94, 68, 15, 41, 42, 39, 37, 63, 98, 90, 39, 2, 13, 31, 28, 57, 4, 71]

]

console.log('numbers by variant: ', numbers)

rl.question('enter number of row you want to handle (all rows - enter \*): ', function (number) {

const transformedMatrix = transformMatrixByCriteria(numbers, number);

let row = []

if(number === '\*'){

row = [...numbers[0], ...numbers[1], ...numbers[2]]

}else{

row = numbers[Number(number) - 1]

}

pareto(transformedMatrix, row)

slater(transformedMatrix, row)

});

rl.on('close', function () {

console.log('\nlog out from the program');

process.exit(0);

});

}

main();