# Завдання на роботу

A close-up of a document

Description automatically generated

# Про реалізацію

Мурашиний алгоритм було реалізовано мовою C#.

Посилання на гітхаб репозиторій: [Bohdan628318ylypchenko/AntColonyOptimization](https://github.com/Bohdan628318ylypchenko/AntColonyOptimization)

Реалізація складається з 3 частин:

1. ant-core – бібліотека, що безпосередньо реалізує мурашиний алгоритм.
2. ant-test – юніт-тести бібліотеки ant-core
3. ant-demo – консольний застосунок, виконує мурашиний алгоритм на заданому графі із вказаними параметрами:

A screenshot of a computer program

Description automatically generated

Рис. 1 Параметри демонстраційного застосунку

Псевдокод реалізації:

1. place ants into start vertex

2. for (iteration count = 0; iteration count < max iteration count; iteration count++):

3. run each ant though graph

4. calculate ferment delta for each graph edge

5. update ferments

6. select best route from current run

7. compare current best and global best

Кроки 1, 2, 4, 5, 6, 7 реалізовані класом AntColony. Крок 3 реалізується класом Ant. Граф, шлях мурахи реалізовані класами Graph, Path відповідно.

Варто відмітити, що під час завантаження графу із файлу всі значення вагів ребер, що рівні нулю, замінюються на велике число: кількість вершин графу \* найбільший ваговий коефіцієнт ребра \* 100. Таким чином, навіть якщо мураха зайде в «глухий кут» (тобто початковий граф є неповним), мураха зможе вийти з кута, пройшовши «неіснуючим» ребром. При цьому включення неіснуючих ребер до маршруту призводить до катастрофічного зростання вартості такого маршруту. Маршрут з неіснуючими ребрами завжди матиме більшу вартість у порівнянні з маршрутом, що включає лише справжні ребра.

# 3. Тестування

## Власний приклад (8 вершин)

В якості найпростішого прикладу для тестування було створено граф:

A diagram of a network

Description automatically generated

Матриця вагових коефіцієнтів вершин (пропуски відповідають значенню 0, будуть замінені на 7 \* 8 \* 100 = 5600):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 |  | 2 | 1 | 3 | 6 |  |  |  |
| 2 | 2 |  |  |  |  | 5 | 7 |  |
| 3 | 1 |  |  |  |  | 4 | 1 |  |
| 4 | 3 |  |  |  |  | 2 | 3 |  |
| 5 | 6 |  |  |  |  | 3 | 3 |  |
| 6 |  | 5 | 4 | 2 | 3 |  |  | 2 |
| 7 |  | 7 | 1 | 3 | 3 |  |  | 1 |
| 8 |  |  |  |  |  | 2 | 1 |  |

Початкове положення – вершина 1. Кінцеве положення – вершина 8. Найкращий шлях із вершини 1 у вершину 8: [1, 3, 7, 8], ціна шляху дорівнює 3.

Оберемо наступні значення параметрів алгоритму:

initial-ferment 0.1

cost-coefficient 1

ferment-coefficient 1

vertex-index-start 0

vertex-index-end 7

ant-count 10

max-iteration-count 10

ant-ferment-count 1

ferment-expiration 0.9

Результат виконання алгоритму:

A screenshot of a computer

Description automatically generated

Реалізація знайшла найдешевший шлях із вершини 1 до вершини 8 (програма нумерує вершини з нуля).

## Власний приклад (13 вершин, 1)

A diagram of a triangle with lines and dots

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 1 |  | 6 | 1 | 1 | 3 |  |  |  |  |  |  |  |  |
| 2 | 6 |  |  |  |  | 1 |  |  |  |  |  |  | 1 |
| 3 | 1 |  |  |  |  | 3 | 1 | 2 |  |  |  |  |  |
| 4 | 1 |  |  |  |  |  |  | 3 | 2 | 1 |  |  |  |
| 5 | 3 |  |  |  |  |  |  |  | 1 | 1 |  |  | 4 |
| 6 |  | 1 | 3 |  |  |  | 2 |  |  |  | 3 |  | 2 |
| 7 |  |  | 1 |  |  | 2 |  | 1 |  |  | 4 |  |  |
| 8 |  |  | 2 | 3 |  |  | 1 |  |  |  | 1 |  |  |
| 9 |  |  |  | 2 | 1 |  |  |  |  |  | 3 | 2 | 4 |
| 10 |  |  |  | 1 | 1 |  |  |  |  |  |  | 2 |  |
| 11 |  |  |  |  |  | 3 | 4 | 1 | 3 |  |  |  | 1 |
| 12 |  |  |  |  |  |  |  |  | 2 | 2 |  |  | 2 |
| 13 |  | 1 |  |  | 4 | 2 |  |  | 4 |  | 1 | 2 |  |

Початкове положення – вершина 1. Кінцеве положення – вершина 13.

Найкращий шлях із вершини 1 у вершину 13: [1, 3, 8, 11, 13]. Ціна шляху дорівнює 5.

Оберемо наступні значення параметрів алгоритму:

initial-ferment 0.1

cost-coefficient 1

ferment-coefficient 1

vertex-index-start 0

vertex-index-end 12

ant-count 10

max-iteration-count 10

ant-ferment-count 1

ferment-expiration 0.9

Результат виконання алгоритму:

A screenshot of a computer

Description automatically generated

Реалізація знайшла найкращий шлях.

## Власний приклад (13 вершин, 2)

Змінимо попередній граф, «замаскувавши» найдешевший шлях. Тепер граф має багато шляхів, ціна яких близька до 5 (6 або 5.5). Відволікаючі розв’язки позначено синім кольором, змінені вагові коефіцієнти – зеленим. Тепер найкращим шляхом є [1, 3, 7, 8, 11, 13].

A diagram of a triangle with lines and dots

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 1 |  | **1** | **2** | 1 | 3 |  |  |  |  |  |  |  |  |
| 2 | **1** |  |  |  |  | 1 |  |  |  |  |  |  | **6** |
| 3 | **2** |  |  |  |  | 3 | **0.5** | 2 |  |  |  |  |  |
| 4 | 1 |  |  |  |  |  |  | 3 | 2 | 1 |  |  |  |
| 5 | 3 |  |  |  |  |  |  |  | 1 | 1 |  |  | 4 |
| 6 |  | 1 | 3 |  |  |  | 2 |  |  |  | 3 |  | **5** |
| 7 |  |  | **0.5** |  |  | 2 |  | 1 |  |  | 4 |  |  |
| 8 |  |  | 2 | 3 |  |  | 1 |  |  |  | **0.5** |  |  |
| 9 |  |  |  | 2 | 1 |  |  |  |  |  | 3 | 2 | 4 |
| 10 |  |  |  | 1 | 1 |  |  |  |  |  |  | 2 |  |
| 11 |  |  |  |  |  | 3 | 4 | **0.5** | 3 |  |  |  | 1 |
| 12 |  |  |  |  |  |  |  |  | 2 | 2 |  |  | 2 |
| 13 |  | **6** |  |  | 4 | **5** |  |  | 4 |  | 1 | 2 |  |

Параметри алгоритму візьмемо ті самі, що і у попередньому прикладі:

initial-ferment 0.1

cost-coefficient 1

ferment-coefficient 1

vertex-index-start 0

vertex-index-end 12

ant-count 10

max-iteration-count 10

ant-ferment-count 1

ferment-expiration 0.9

Результат виконання алгоритму:

A screenshot of a computer

Description automatically generated

Реалізація знайшла найкращий шлях.

## SHP 55,95,155

У попередніх прикладах мурашиний алгоритм не потребував ретельного підбору параметрів для знаходження найкращого шляху. Тому під час дослідження графів shp 55, 95, 155 хотілось би провести декілька запусків, змінюючи параметри таким чином, що буде наявне поступове покращення результату.

Набір параметрів для графу shp55:

# Greedy 1

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 10 -i 100 -q 100 -c 0.9

# Greedy 2

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 10 -i 200 -q 100 -c 0.9

# Greedy 3

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 10 -i 300 -q 100 -c 0.9

# Greedy 4

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 10 -i 400 -q 100 -c 0.9

# Greedy 5

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 50 -i 400 -q 100 -c 0.9

# Greedy 6

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 50 -i 500 -q 100 -c 0.9

# Greedy 7

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 0 -s 0 -e 54 -t 50 -i 1000 -q 100 -c 0.9

# Ferments 1

..\ant-demo\bin\x64\Release\net8.0-windows\ant-demo.exe -r "./normalized-quantum-big.rnd" -p "./SHP55.aco" -f 0.1 -a 1 -b 1 -s 0 -e 54 -t 50 -i 400 -q 100 -c 0.9

Перші 7 досліджень встановлюють параметр b (ferment-coefficient) рівним 0. Тобто для перших 7 досліджень вплив ферменту на поведінку алгоритму відсутній. Тоді поведінка алгоритму має деяку схожість із жадібним алгоритмом: імовірність вибору вершини А залежить виключно від ціни шляху між поточною вершиною та вершиною А.

Останнє дослідження встановлює коефіцієнт ціни та коефіцієнт ферменту рівними 1.

Дослідження також змінюють кількість ітерацій (від 100 до 1000), кількість мурах (10, 50).

Для графів shp 95, 155 параметри варіювались за аналогічним принципом.

Результати досліджень графу shp55:

A screenshot of a computer

Description automatically generated

Результати досліджень графу shp95:

A screenshot of a computer

Description automatically generated

Результати досліджень графу shp155:

A screenshot of a computer

Description automatically generated

Для кожного графу всі дослідження знаходять однаковий шлях, вплив ферментів не має значення:

Граф shp55: [0, 1, 2, 54], ціна = 3

Граф shp95: [0, 1, 3, 5, 94], ціна = 12

Граф shp155: [0, 1, 3, 5, 7, 154], ціна = 5

Імовірно, ці шляхи є найкращими.

Здатність алгоритму знаходити найкращий шлях для графів shp55,95,155 без впливу ферментів («жадібна імовірність») може бути пов’язана з великим розкидом вагів графу:

A close up of a screen

Description automatically generated

Приклад значень вагів графу shp55

Наявні малі значення (1, 4, 3, 46, 40, 35 тощо) і великі значення (210, 265, 189 тощо). Такий розмах значно полегшує пошук шляху, адже найкращий шлях додатково «підсвітчується» ребрами з великою вагою: для знаходження найкращого шляху достатньо уникати ребра з великою вагою.

Подальша оптимізація параметрів не має сенсу: алгоритм знаходить оптимальні шляхи за задовільний час (від 100/200 мілісекунд до 4 секунд для графу shp155 з 1000 ітераціями).