**

***Міністерство освіти і науки України***

*Національний технічний університет України*

*«Київський політехнічний інститут імені Ігоря Сікорського»*

***Кафедра АСОІУ***

***Спеціальність 6,050101 «Комп’ютерні науки»***

**КУРСОВА РОБОТА**

**з дисципліни**

**“Додаткові розділи дослідження операцій”**

**на тему**

«Задача Ноя»

|  |  |  |
| --- | --- | --- |
| *Виконали*  студент групи ІС-63  Березінський Г.В.  N зал. кн. ІС-6302  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис)  студент групи ІС-63  Ільєнко Р.С.  N зал. кн. ІС-6309  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка) | *Прийняли*  Доц. каф. АСОІУ, к.т.н.  ЖДАНОВА О.Г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис)  Доц. каф. АСОІУ, к.т.н.  ЖУРАКОВСЬКА О.С.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис)  Доц. каф. АСОІУ, к.т.н.  СПЕРКАЧ М.О.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) |

**Київ-2019**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Національний технічний університет України*  *«Київський політехнічний інститут імені Ігоря Сікорського»* | | | | | | | | | | | | | |
| (назва вищого навчального закладу) | | | | | | | | | | | | | |
| Кафедра | АСОІУ | | | | | | | | | | | | |
| Дисципліна | | Додаткові методи дослідження операцій | | | | | | | | | | | |
| Спеціальність | | | | ***122 Комп'ютерні науки та інформаційні технології*** | | | | | | | | | |
| Курс | | | | | 3 | | | Група | | ІС-63 | | Семестр | 6 |
| **Завдання**  НА КУРСОВУ РоБОТУ СТУДЕНТУ | | | | | | | | | | | | | |
| ***Березінський Геннадій Валерійович***  ***Ільєнко Роман Сергійович*** | | | | | | | | | | | | | |
| 1. Тема роботи | | | *Задача Ноя* | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 2. Термін здачі студентом закінченої роботи | | | | | | | | | *30 травня 2019 року* | | | | |
| 3. Завдання курсової роботи | | | | | | |  | | | | | | |
| *Вид робіт* | | | | | | | | | | | *Виконавець* | | |
| *Побудувати математичну модель задачі.* | | | | | | | | | | | *Березінський Г., Ільєнко Р.* | | |  |
| *Виконати аналіз можливих методів розв’язання* | | | | | | | | | | | *Березінський Г., Ільєнко Р.* | | |
| *Розробити алгоритм муравїних колоній* | | | | | | | | | | | *Березінський Г* | | |
| *Розробити генетичний алгоритм* | | | | | | | | | | | *Ільєнко Р.* | | |
| *Виконати програмна реалізація алгоритму муравїних колоній.* | | | | | | | | | | | *Березінський Г* | | |
| *Програмна реалізація генетичного алгоритму* | | | | | | | | | | | *Ільєнко Р.* | | |
| *Визначити складність розроблених алгоритмів (теоретичну та експериментальну* | | | | | | | | | | | *Березінський Г., Ільєнко Р.* | | |
| *Програмно реалізувати генератор індивідуальних задач (ГІЗ)* | | | | | | | | | | | *Березінський Г., Ільєнко Р.* | | |
| *Використовуючи ГІЗ, виконати порівняльний аналіз розроблених алгоритмів* | | | | | | | | | | | *Березінський Г., Ільєнко Р.* | | |
| 4. Зміст розрахунково-пояснювальної записки | | | | | | | | | | | | | |
| *Змістовні постановки задачі. Математична постановка задачі.* | | | | | | | | | | | | | |
| *Аналіз можливих методів розв’язання. Опис розроблених алгоритмів розв’язання задачі.* | | | | | | | | | | | | | |
| *Опис програмної реалізації розроблених алгоритмів. Приклади застосування алгоритмів.* | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 5. Перелік графічного матеріалу (с точним зазначенням обов’язкових креслень) | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 6. Дата видачі завдання | | | | | | *20 лютого 2019 року* | | | | | | | |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Назва етапів виконання курсової роботи** | | | **Строк виконання етапів роботи** | **Примітка** |
| 1 | *Опис постановки задачі* | | | *01.03* |  |
| 2 | *Консультація у керівника* | | | *05.03* |  |
| 3 | *Пошук літературних джерел за темою курсової роботи. Порівняльний аналіз існуючих методів розв’язання задачі* | | | *15.03* |  |
| 4 | *Вивчення теоретичних положень генетичного алгоритму та алгоритму муравїних колоній* | | | *31.03* |  |
| 5 | *Розробка генетичного алгоритму та алгоритму муравїних колоній* | | |  |  |
| 6 | *Розв’язання задач “вручну”* | | |  |  |
| 7 | *Консультація в керівника – погодження інтерфейсу програми* | | |  |  |
| 8 | *Розробка програмної реалізації. Відлагодження програм. Визначення найбільш ефективної комбінації варіантів основних алгоритмів* | | |  |  |
| 9 | *Оцінка складності алгоритмів розв’язання задачі* | | |  |  |
| 10 | *Оформлення пояснювальної записки* | | |  |  |
| 11 | *Захист курсової роботи* | | |  |  |
|  | | | | | |
| Студенти | |  | *Березінський Г.В.* | | |
|  | | (підпис) | (прізвище, ініціали) | | |
|  | |  | *Ільєнко Р.С.* | | |
|  | | (підпис) | (прізвище, ініціали) | | |
| Керівник | |  |  | | |
|  | | (підпис) | (прізвище, ініціали) | | |
| «*20*» *лютого 2019 року* | | | | | |

# Розділ 1. ПОСТАНОВКА ЗАДАЧІ

Існує N видів тварин (Т1, Т2 … Тn). Для цих тварин задана матриця сумісностей, у якій в клітинці aij стоїть 1, якщо тварина i може бути в одному приміщенні з твариною j, інакше – 0. Яких тварин необхідно взяти, щоб їх кількість була найбільша.

* 1. **Індивідуальна постановка задач**

Існує п’ять видів тварин (Т1, Т2, Т3, Т4, Т5). Для розміщення тварини у одному приміщенні, вони повинні бути сумісними один з одним. i-та та j-та тварина сумісна одна з одною якщо у матриці сумісності у клітинці ij стоїть 1, якщо ні - 0. Кожна тварина має коефіцієнт корисності, який рівень одиниці.

Знайти яких тварин необхідно взяти Ною аби їх число було максимальним.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Т1 | Т2 | … | Тn |
| Т1 | a11 | a12 | … | a1n |
| Т2 | a21 | a22 | … | a2n |
| … | … | … | … | … |
| Т4 | an1 | an2 | … | ann |

**Індивідуальна постановка 1. Березінський Г.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Т1 | Т2 | Т3 | Т4 | Т5 |
| Т1 | 1 | 0 | 1 | 1 | 0 |
| Т2 | 0 | 1 | 1 | 0 | 1 |
| Т3 | 1 | 1 | 1 | 0 | 1 |
| Т4 | 1 | 0 | 0 | 1 | 0 |
| Т5 | 0 | 1 | 1 | 0 | 1 |

Допустимі розв’язки:

R1 = {1, 3};

R2 = {2, 3, 5}

**Індивідуальна постановка 2. Ільєнко Р.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Т1 | Т2 | Т3 | Т4 | Т5 |
| Т1 | 1 | 1 | 0 | 0 | 1 |
| Т2 | 1 | 1 | 0 | 1 | 0 |
| Т3 | 0 | 0 | 1 | 1 | 0 |
| Т4 | 0 | 1 | 1 | 1 | 1 |
| Т5 | 1 | 0 | 0 | 1 | 1 |

Допустимі розв’язки:

R1 = {1, 2};

R2 = {5, 4}

**Математична модель постановки задачі**

**Змінні:**

Xi = 1, якщо тварину беруть, інакше – 0.

**ЦФ:**

Цільова функція – максимізація кількості тварин на ковчегу.

**Обмеження:**

Необхідно знайти, яких тварин можна взяти для того, щоб їх кількість була максимальною.

i = (1 … n), j = (1 … n), i ≠ j; (1),

Обмеження (1) гарантує, що на ковчег не потраплять не сумісні тварини.

Розділ 2. Опис методів розв’язання задачі

**2.1. Існуючі методи розв’язання.**

Виходячи з класу задачі можна використовувати наближені алгоритми розв’язку такі як генетичні алгоритми, алгоритм мурашиних колоній, бджолиний і тд..

**2.2. Розробка генетичного алгоритму**.

Генетичний алгоритм це в першу чергу еволюційний алгоритм, що заснований на схрещуванні(комбінуванні). Шляхом перебору та відбору отримаємо правильну комбінацію. Алгоритм поділяється на три етапи:

* Схрещування
* Селекція(відбір)
* Формування нового покоління

Якщо результат нас не влаштовує, кроки повторюються допоки результат нас не влаштує або якщо кількість поколінь(повторів) досягне свого максимуму.

Опис алгоритму:

1. Випадковим чином генерується кінцевий набір готових рішень:

, (перше покоління, n – розмір популяції)

1. Оцінка пристосовуваності поточного покоління

1. Вихід, якщо виконується критерій зупинки.
2. Генерація нового покоління S, схрещуючи C та мутацій

та перехід до пункту 2 в іншому випадку.

В процесі селекції відбираються лише декілька кращих пробних рішень, інші далі не використовуються. Схрещування за місце пари рішень створює іншу, елементи якої перемішані випадковим чином. Мутація випадковим чином змінює певну компоненту рішення на нуль.

Опис алгоритму:

Вхід: А – матриця сумісності тварин

m – розмір початкової популяції

α – ймовірність мутації

F – умова завершення, впродовж 20 ітерацій алгоритм не призводив до покращення розв’язку

Крок 1. Генерація початкової популяції

На першому кроці генерується популяція з m членів, в якій тварини, що входять до неї обираються ввипадковим чином. Алгорим перевіряє популяцію та визначає членів, які не задовільняють умові.

while не виконується F do

Крок 2. Обрати батьків.

Для вибору батьків скористаємося турнірним методом, в якому кожному популяція буде поділена на дві частини випадковим чином. Кожному члену буде присвоєно номер команди (1 або 2). З кожної команди буде обрано одного найкращого представника. Так, у результаті ми матимемо двох батьків та можемо переходити до створення нащадків.

Крок 3. Схрещення.

На етапі схрещення отриманих у попередньому кроці батьків ми застосуємо одноточковий кросинговер. Для цього визначимо опорну точку кросинговеру, поділивши порівну гени на дві частини. Далі формуємо двох нащадків. Випадковим чином обирається точка кросинговеру. Перший нащадок отримає гени першого батька по ліву частину точки та другого по праву, другий- праву частину першого та ліву другого.

Крок 4. Мутації.

На цьому етапі, з певною ймовірністю з кожним отриманим нащадком може відбутися мутація. Використаємо одноточковий оператор мутації. Приймемо ймовірність мутації за 0,1. При мутації, випадковий ген нащадка змінює своє значення.

Крок 5. Локальне покращення(Реанімація)

Якщо один з нащадків виявився непридатним, спробуємо зробити реанімацію. Оцінимо, які тварина конфліктують одна з одною. Оцінимо, з якою кількістю конфліктує кожна тварина. Далі, поки не буде досягнуто допустимого розв’язку, на кожній ітерації прибираємо тварину, яка має найбільше конфліктів, після чого перераховуємо конфлікти, і, якщо їх немає, переходимо до включення в популяцію.

Крок 6. Оновлення популяції.

При оновленні популяції слід робити відсів 2-х найгірших розв’язків, щоб не працювати з поганими результатами. Для цього додамо отриманих нащадків в популяцію та оцінимо цільову функцію (кількість тварин в кожному наборі) для кожного члена популяції. Два найгірших результати видаляються з популяції. Якщо один з отриманих результатів є кращим за попередній рекорд, визначаємо новий рекорд. Якщо покращення не було впродовж 20 ітерацій, завершуємо роботу алгоритму.

**2.3. Опис методу розв’язання**

Алгоритм 1 будується на випадковому відборі популяції, схрещуванні представників популяції з метою отримати краще потомство.

Алгоритм 1 завжди призведе до допустимого розв’язку так як на одному з кроків відбувається перевірка, чи задовольняє нащадок умові задачі. Отже, алгоритм буде відсіювати недопустимі розв’язки. Метод є наближеним, тому точної гарантії на отримання оптимального розв’язку немає. Алгоритм завершує роботу, після того, як впродовж 10 ітерацій не було покращення ЦФ.

Генетичний алгоритм працює за лінійний час та має лінійну складність.

**2.4. Опис алгоритму мурашиних колоній.**

Алгоритм мурашиних колоній представляє собою жадібний еврістичний алгоритм, що будує нове рішення на основі попереднього. Основною ідеєю є моделювання поведінки мурах на шляху до їжі. Мурахи позначають кращі шляхи феромонами, по котрих ідуть наступні мурахи. В умовах даної задачі позначатимемо «феромоном» тварин, які можуть бути перевезені з найбільшою кількістю інших тварин. На заключних кроках алгоритм будуватиме популяції переважно з цих тварин.

Алгоритм завжди приводитиме до допустимого так як використовуються графи та мурахи можуть переходити лише до суміжних вершин.

Алгоритм є наближеним, тому не гарантує оптимальність розв’язку.

*Опис алгоритму:*

Крок 0. Ініціалізація початкових параметрів.

Початкове значення феромону – 1. Кількість мурах – 1.

α = 1

β = 1

ρ = 0.4

Доки не досягнуто умов виходу виконуються наступні кроки:

Крок 1. Створення мурах.

На першому кроці кожної ітерації розміщаємо всю колонію у випадкову вершину. Ймовірність розміщення колонії у вершині вираховується за формулою:

де – кількість шляхів з вершини і, N – кількість ребер. Тобто ймовірність розміщення колонії у вершині прямо пропорційна кількості суміжних з нею вершин. Поточну вершину включимо у шлях S

Крок 2. Пошук рішень.

Вирахуємо ймовірності переходу мурахи у суміжну вершину. Ймовірність переходу у і-ту суміжну вершину:

(1)

Де – кількість спільних суміжних вершин з вершиною і та вершин зі шляху S, - значення феромону для даного шляху.

Після переходу додамо вершину у шлях та за формулою (1) вирахуємо наступний перехід. Цю операцію повторюватимемо доки не залишиться вершин, що суміжні з усіма вершинами шляху.

Коли знайдено всі вершини шляху, якщо поточний шлях кращий на поточний кращий, оновлюємо його.

Крок 3. Оновлення феромонів.

Перерахуємо феромони. Для кожного ребра, що входить в шлях нове значення феромону

Де L – кількість вершин у знайденому шляху, N – вершин всього.

В іншому випадку:

Якщо існують вершини, що зв’язані лише з однією суміжною, помітимо її як не оптимальну перерахувавши феромон як

Кроки 1-3 повторюються, доки вподовж 20 ітерацій не буде покращення результату.

Алгоритм є еврістичним, буде зроблено m (m ≥ 20) ітерацій, кожна з яких є лінійною.

**2.5. Приклади застосування розроблених алгоритмів.**

Нехай задача задана матрицею

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |

* *Приклад застосування генетичного алгоритму.*

На першому кроці генеруємо популяцію з 8 членів. Позначимо ЦФ як .

1. (100101) , НЕ ЗАДОВІЛЬНЯЄ
2. (011001) , НЕ ЗАДОВІЛЬНЯЄ
3. (011000)
4. (010101) НЕ ЗАДОВІЛЬНЯЄ
5. (010010)
6. (001011)
7. (110001) НЕ ЗАДОВІЛЬНЯЄ
8. (111000)

Виключимо з популяції членів, що не задовільняють умові.

Для вибору батьків назначимо турнір між 3 та 8 і 5 та 6.

6 та 8 є кандидатами в батьки.

За допомогою одноточкового кросинговеру маємо двох нащадків. Випадковою точкою є точка 3.

1. (001000)
2. (111011)

З ймовірністю α кожен нащадок мутує, його випадковий ген інвертується. Обидва нащадки не піддаються мутації.

Перший нащадок задовольняє умові. Другий – ні. Проведемо для нього реанімацію. Для цього доки розв’язок не стане допустимим, прибиратимемо найконфліктнішу тварину. Для другого нащадка мажмо, що 1 та 2 тварина мають по одному конфлікту(з 6) 3 та 5 – не має жодного, а 6 – конфліктує з 2. Прибираємо її як найконфліктнішу.

Змінений нащадок має вигляд:

(111010)

Так як цей розв’язок є кращим, за поточний найкращий, запишемо його, як поточний найкращий. Тобто ми отримали покращення розв’язку.

Додамо наших нащадків до популяції та приберемо з популяції 2 найгірші розв’язки.

Нова популяція матиме вигляд:

1. (010010)
2. (001011)
3. (111000)
4. (111010)

Повторюватимемо алгоритм, доки впродовж 20 ітерацій не буде покращення розв’язку. У цьому випадку, алгоритм повертає поточне найкраще значення як (наближено) оптимальний розв’язок.

* *Приклад застосування мурашиного алгоритму.*

Маємо наступну задачу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |

Що може бути представлена наступним графом

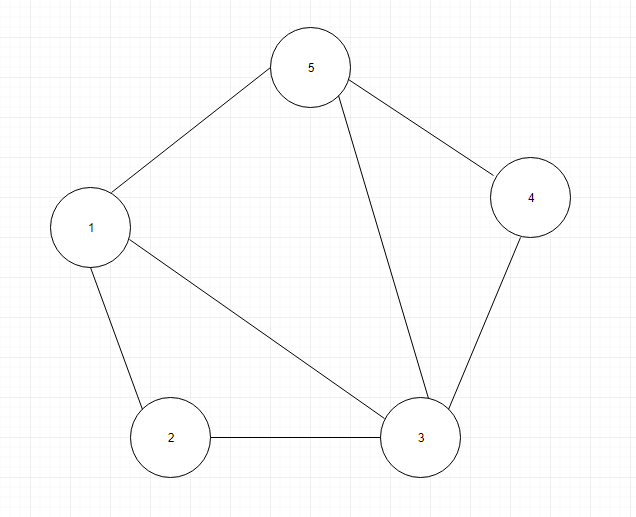
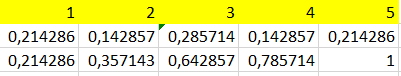


Рисунок 2.1 - Зображення задачі у вигляді графу.

Ймовірності переходів та відповідних діапазонів представлені



Випадає число 0,57, початкова вершина – 3.

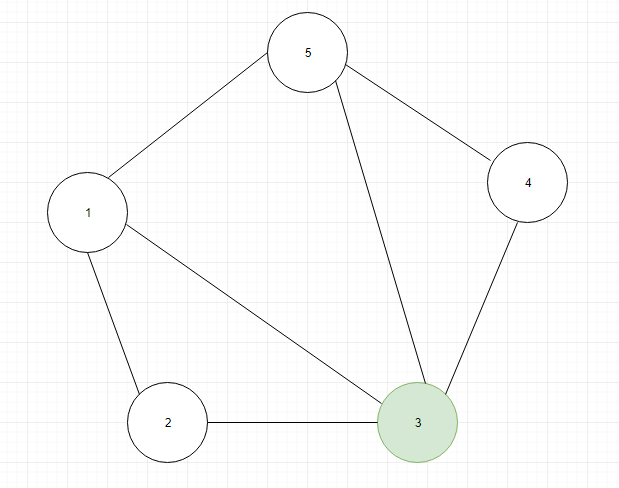
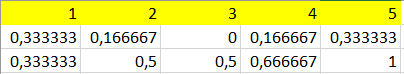


Рисунок 2.2 - Крок алгоритму.

Ймовірності переходів та відповідні діапазони представлені нижче:



Випадає число 0,21. Переходимо у 1 вершину.

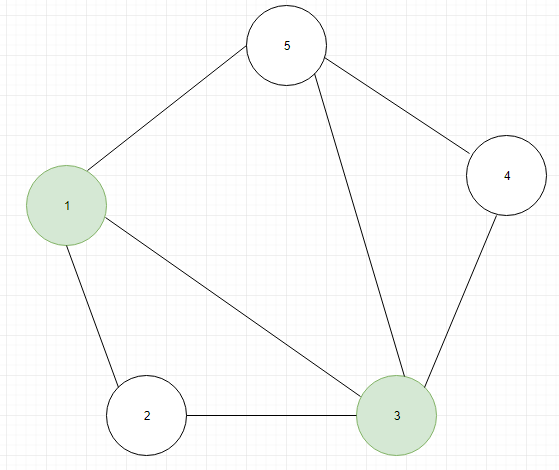
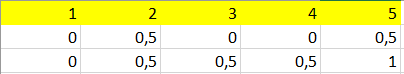


Рисунок 2.3 - Крок алгоритму.

Перераховуємо ймовірності



4 вершина 0 так як . Тобто 4 вершина не суміжна кожному елементу поточного шляху.

Випадає число 0,37. Переходимо у вершину 2.

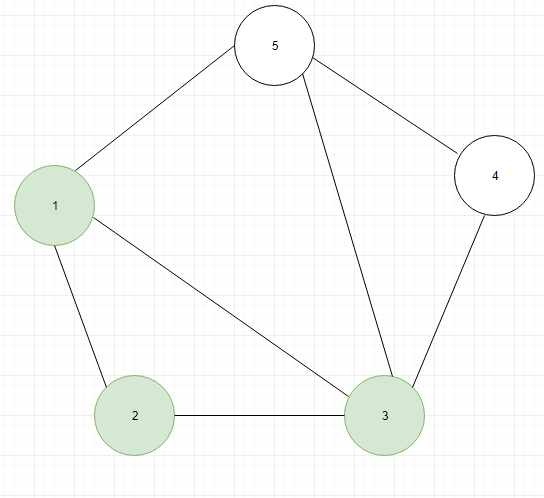


Рисунок 2.4 - Крок алгоритму.

Перша ітерація завершена так як усі

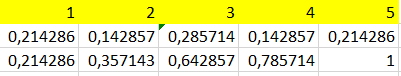
Перерахуємо феромони нова матриця феромонів матиме вигляд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1,36 | 1,36 | 0 | 0,4 |
| 1,36 | 0 | 1,36 | 0 | 0 |
| 1,36 | 1,36 | 0 | 0,4 | 0,4 |
| 0 | 0 | 0,4 | 0 | 0,4 |
| 1,36 | 0 | 0,4 | 0,4 | 0 |

L = 3.

Ітерація 2.

Ймовірності переходів та відповідних діапазонів представлені



Випадає число 0,84, початкова вершина – 4.

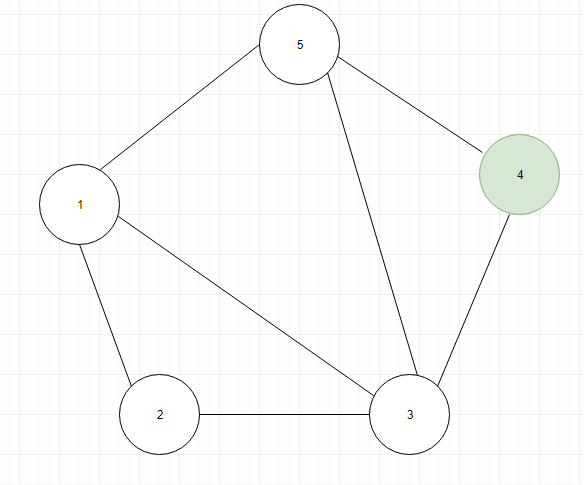


Рисунок 2.5 - Крок алгоритму.

Розрахуємо ймовірності подальшого переходу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0,5 | 0 | 0,5 |
| 0 | 0 | 0,5 | 0,5 | 1 |

Випадає число 0,40. Переходимо у вершину 3.

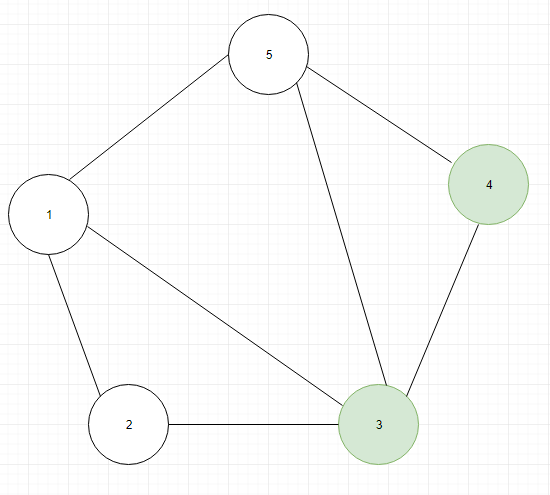


Рисунок 2.6 - Крок алгоритму.

Зі ймовірністю 1 переходимо у вершину 5, як єдину, що сумісна з кожним елементом шляху.

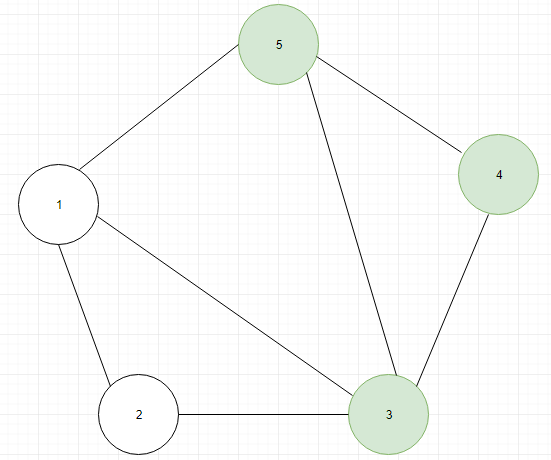


Рисунок 2.7 - Крок алгоритму.

L = 3. Розв’язок не покращився, перерахуємо феромони.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0,544 | 0,544 | 0 | 0,16 |
| 0,544 | 0 | 0,544 | 0 | 0 |
| 0,544 | 0,544 | 0 | 0,76 | 0,76 |
| 0 | 0 | 0,76 | 0 | 0,76 |
| 0,544 | 0 | 0,76 | 0,76 | 0 |

Через 18 ітерацій у разі не покращення розв’язку алгоритм припинить свою роботу.

# 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

## **3. 1 Опис функціональної моделі**

Програмний продукт (ПП) містить наступні складові:

1. **робота з індивідуальною задачею** (ІЗ):
   1. введення вручну даних ІЗ;
   2. генерація (випадковим чином) даних ІЗ;
   3. збереження в файл/читання з файлу/редагування даних ІЗ;
   4. розв’язання ІЗ усіма розробленими методами (з виведенням на екран відповідних розв’язків та значень ЦФ);
2. **експериментальне дослідження розроблених алгоритмів:** 
   1. задання діапазонів зміни параметрів експериментів:
      * розмірність задач (від; до; крок);
      * кількість ІЗ, яку необхідно згенерувати для кожної розмірності;
      * діапазоні зміні коефіцієнтів (ЦФ та обмежень);
   2. генерація множини ІЗ;
   3. розв’язання множини згенерованих ІЗ усіма розробленими методами;
   4. виведення результатів експериментів:
      * порівняння за точністю;
      * порівняння за часом.

## **3. 2 Рішення з інформаційного забезпечення**

### 3.2.1 Вхідні дані

На вхід алгоритм отримує матрицю суміжності. Матриця суміжності зчитується з файлу, генерується випадковим чином або вводиться через користувацький інтерфейс (рисунок 3.1).



Рисунок 3.1 - Приклад користувацького інтерфейсу для введення даних.

### 3.2.2 Вихідні дані

На вихід алгоритм видає час виконання у секундах, значення цільової функції z (оптимальну кількість тварин) та шлях – тварин, яких необхідно взяти, щоб їх кількість була найбільшою. Також програма малює граф по вхідним даним та позначає вершини, що входять в шлях (рисунок 3.2).

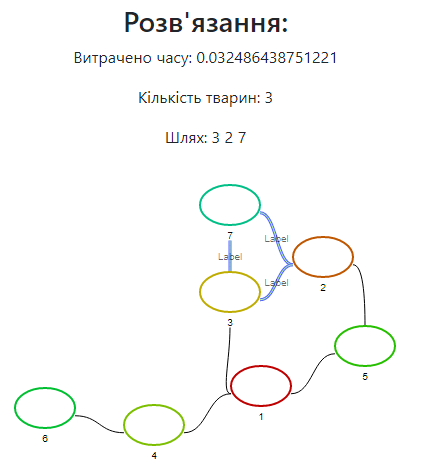


Рисунок 3.2 - Приклад вихідних даних.

### 3.2.3 Опис інформаційного забезпечення

В цьому пункті наводиться опис способів зберігання інформації.

Якщо при реалізації ПП розроблена база даних, то необхідно представити її у вигляді ER-діаграми з описом (або фізичної моделі бази даних). Також опишіть таблиці БД за шаблоном, представленим в таблиці 3.1.

***Таблиця 3.1 – Шаблон опису таблиці БД***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Назва таблиці*** | ***Назва поля*** | ***Тип*** | ***Опис*** |
| algorithmInfo | Id | Int(11) | Використовується для як унікальний ідентифікатор |
| Time | Float | Час виконання алгоритму |
| Animals | Int(11) | Кількість сумісних тварин |
| MatrixSize | Int(11) | Розмір матрці |
| Type | Tinyint(4) | Алгоритм мурашиний чи генетичний |

## **3.3 Рішення з програмного забезпечення**

### 3.3.1 Опис обраних засобів розробки

Усі обчислення виконувались засобами python. Серверна частина написана на php що викликав скрипти python для розрахунків. Фронтенд частина написана з використанням html, css, js, а саме такі фреймворки: bootstrap, dracula graph library, charjs.

### 3.3.2 Архітектура програмного забезпечення

#### 3.3.2.1 Схема архітектури ПП

#### ├── algorithm.php (вивід відомостей про алгоритм)

#### ├── antDescription.php (вивід опису про мурашиний алгоритм)

#### ├── authors.php (вивід відомостей про авторів)

#### ├── css (папка зі стилями)

#### │   └── bootstrap.min.css (стиля фреймворку бутстрап)

#### ├── example.php (вивід результату обчислення)

#### ├── favicon.ico (іконка)

#### ├── functions.php (файл з функціями пхп)

#### ├── geneticDescription.php (вивід відомостей про генетичний алгоритм)

#### ├── header.php (шапка сайту)

#### ├── img (папа з рисунками)

#### ├── index.php (сторінка з основною статистикою)

#### ├── js (папка з джс скриптами)

#### │   ├── bootstrap.min.js (скрипт бутсрапу)

#### │   ├── Chart.min.js (скрипт графіків)

#### │   ├── dracula.min.js (скрипт графів)

#### │   ├── jquery-3.3.1.slim.min.js (джейкуері скрипт)

#### │   ├── popper.min.js (скрипт бутсрапу)

#### │   └── raphael.js (скрипт графів)

#### ├── phpmyadmin -> /usr/share/phpmyadmin (інструмент керування БД)

#### └── system (папка з обчислювальними скриптами)

#### ├── countAlgorithm.php (скрипт що визиває потрібний python файл

#### ├── input.txt (вхідні дані користувача)

#### ├── output.txt (вихідні дані)

#### ├── python (папка з python скриптами)

#### │   ├── antAlgorithm.py (скрипт що обчислює матрицю за домопогою мурашиного алгоритму)

#### │   └── geneticAlgorithm.py (скрипт що обчислює матрицю за домопогою генетичного алгоритму)

#### └── uploadAlgorithm.php (скрипт що опрацьовує завантажені файли)

#### 3.3.2.3 Специфікація функцій

antAlgorithm.php

|  |  |  |
| --- | --- | --- |
| Назва функції | Аргументи | Опис |
| findAnimals | head | Знаходить сумісних тварин з даною вершиною |
| countNumberOfEdges |  | Підраховує кількість ребер в графі |
| countEdgesFromHead | head | Рахує ребра з вершини |
| countStartHeadProbability | numberOfEdges | Вираховує початкові ймовірності переходу для мурашиного алгоритму |
| countHeadProbability | head, commonHeads, headSum | Вираховує ймовірності переходу для мурашиного алгоритму |
| choosHeadToGo | startHeadProbability, randomWay | Обирає куди підти далі |
| countCommonHeads | markedList | Підраховує ребра між вершинами в поміченому списку |

geneticAlgorithm.py

|  |  |  |
| --- | --- | --- |
| Назва функції | Аргументи | Опис |
| returnAnimalsList | animals | Повертає список тварин з якими сумісна дана тварина |
| changeChar | s, p, r | Заміняє символ у строці |
| getRandomPopulation | stringLength | Генерує випадкову популяцію |
| checkAnimals | matrix, animalOneNumber, animalTwoNumber | Перевіряє тварин на сумісність |
| checkIfSatisfy | matrix, item | Перевіряє чи усі члени популяції сумісні |
| countAnimals | item | Підраховує тварин у популяції |
| printPopulations | count, startMatrix, populationItem | Виводить інформацію про популяцію |
| findMaxFromPopulations | firstGroup, secondGroup | Обирає найркащу популяцію з двох груп |
| crossingOver | globalCout, parentsList | Функція кросинговеру |

## **3.4 Керівництво користувача**

### 3.4.1 Інструкція користувача

Система складається з 4-х частин: головна сторінка зі статистикою, мурашиний алгоритм, генетичний алгоритм, відомості про авторів.

На головній сторінці можна побачити графіки залежності часу виконання програми від розмірності матриці (Рисунок 3.3, 3.4).

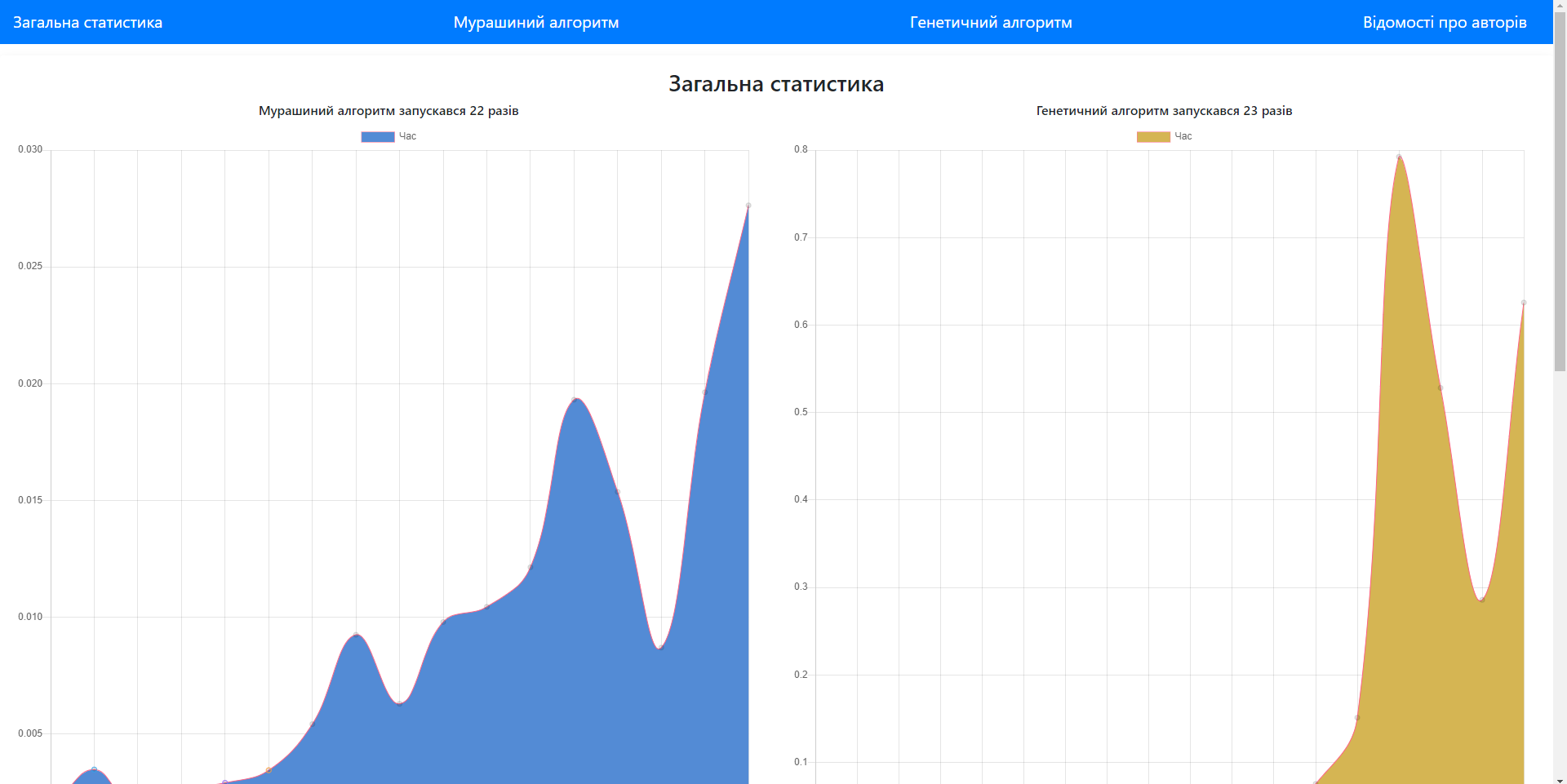


Рисунок 3.3 – Графіки залежності часу виконання від розміру вхідних даних

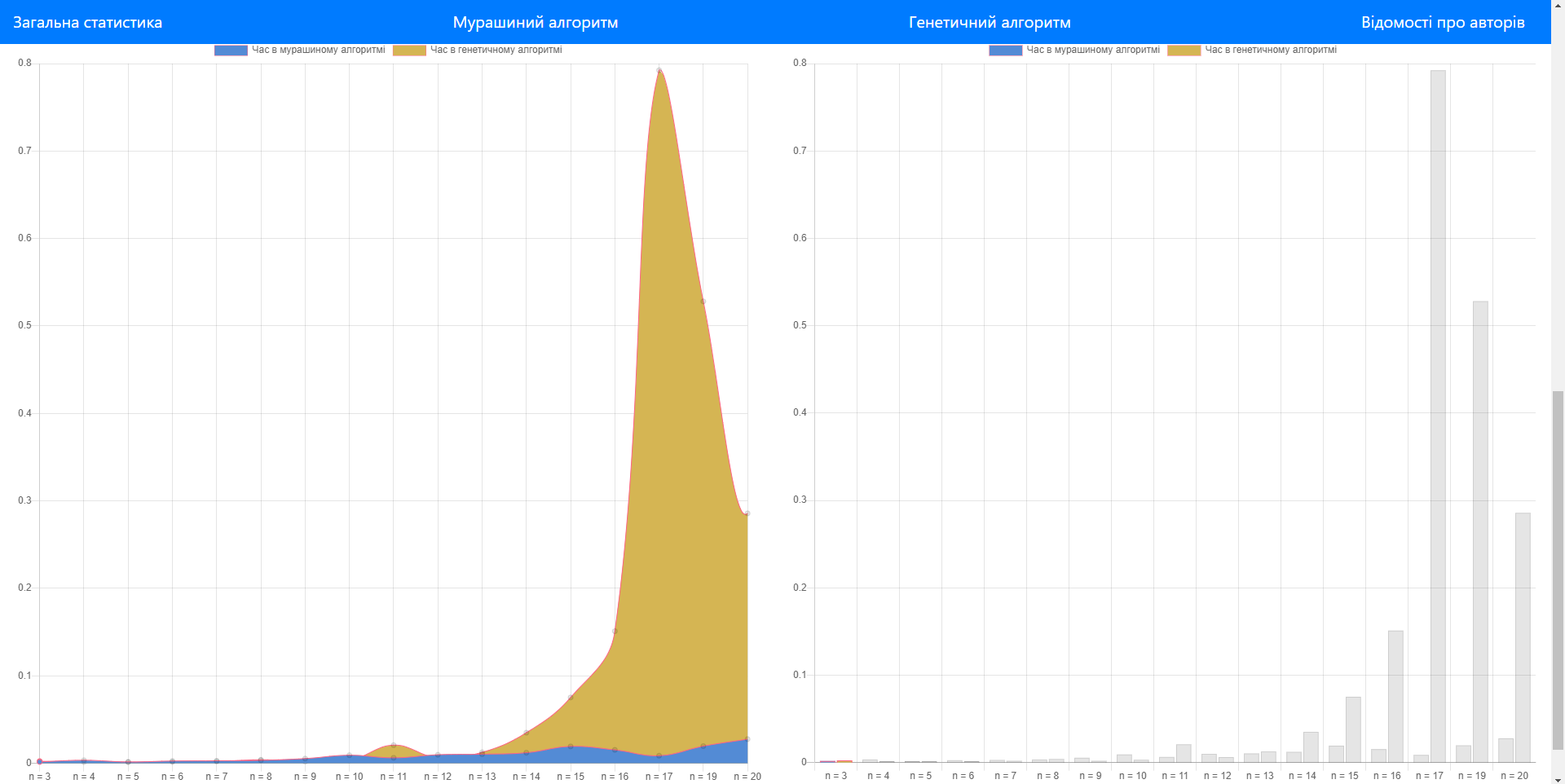


Рис. 3.4 - Графіки залежності часу виконання від розміру вхідних даних

На сторінці мурашиного алгоритму можна прочитати опис методу та обрати розмірність матриці (Рисунок 3.5).

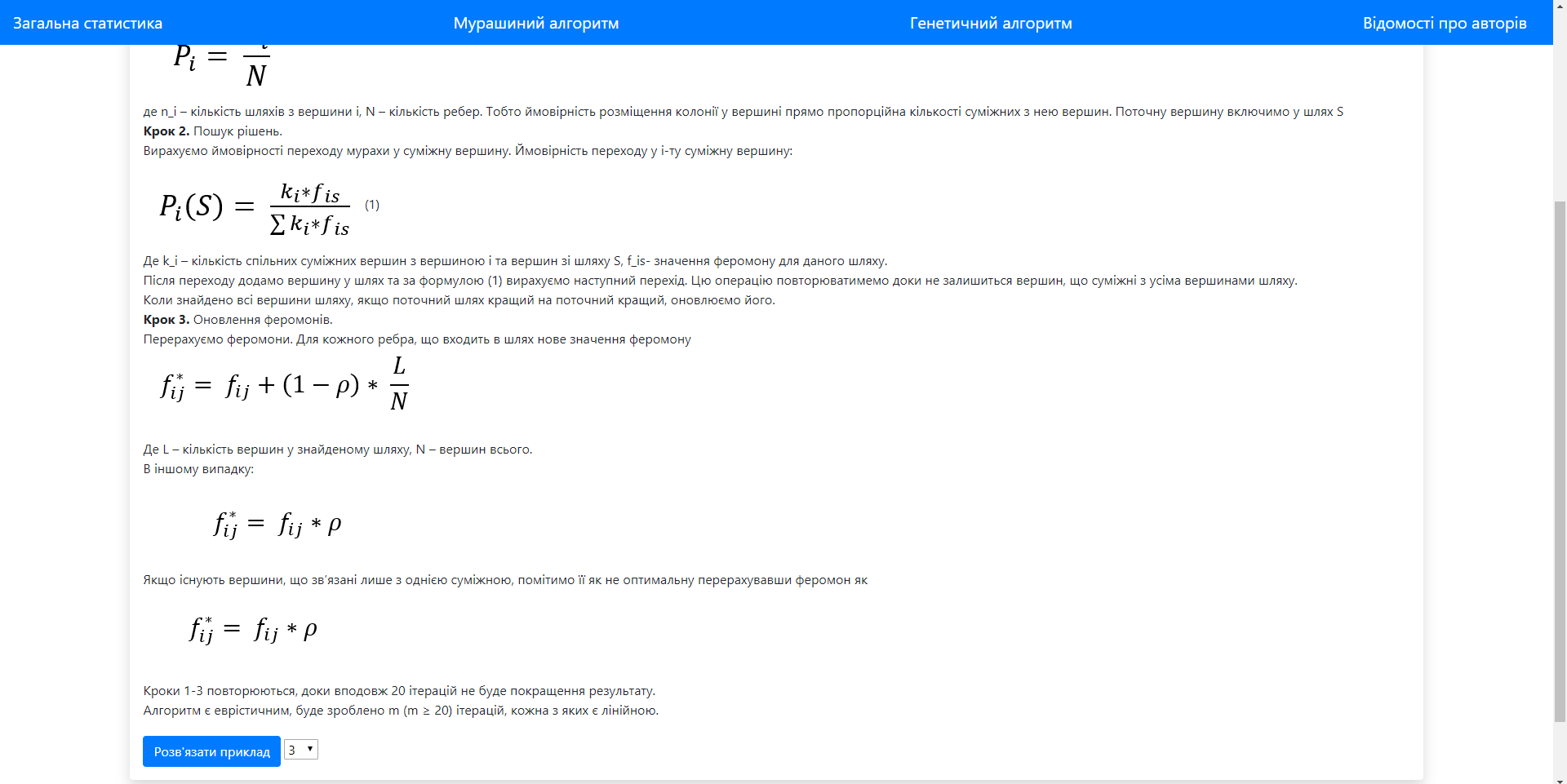


Рисунок 3.5 – стартова сторінка мурашиного алгоритму

Обравши розмірність матриці, користувач переходить на сторінку розв’язку матриці. Можна ввести дані вручну, заповнити випадковим чином або завантажити з файла. Після введення даних, треба натиснути кнопку «Роозв’зати» після чого з’явиться відповідь та граф з потрібними маршрутами (Рисунок 3.6).

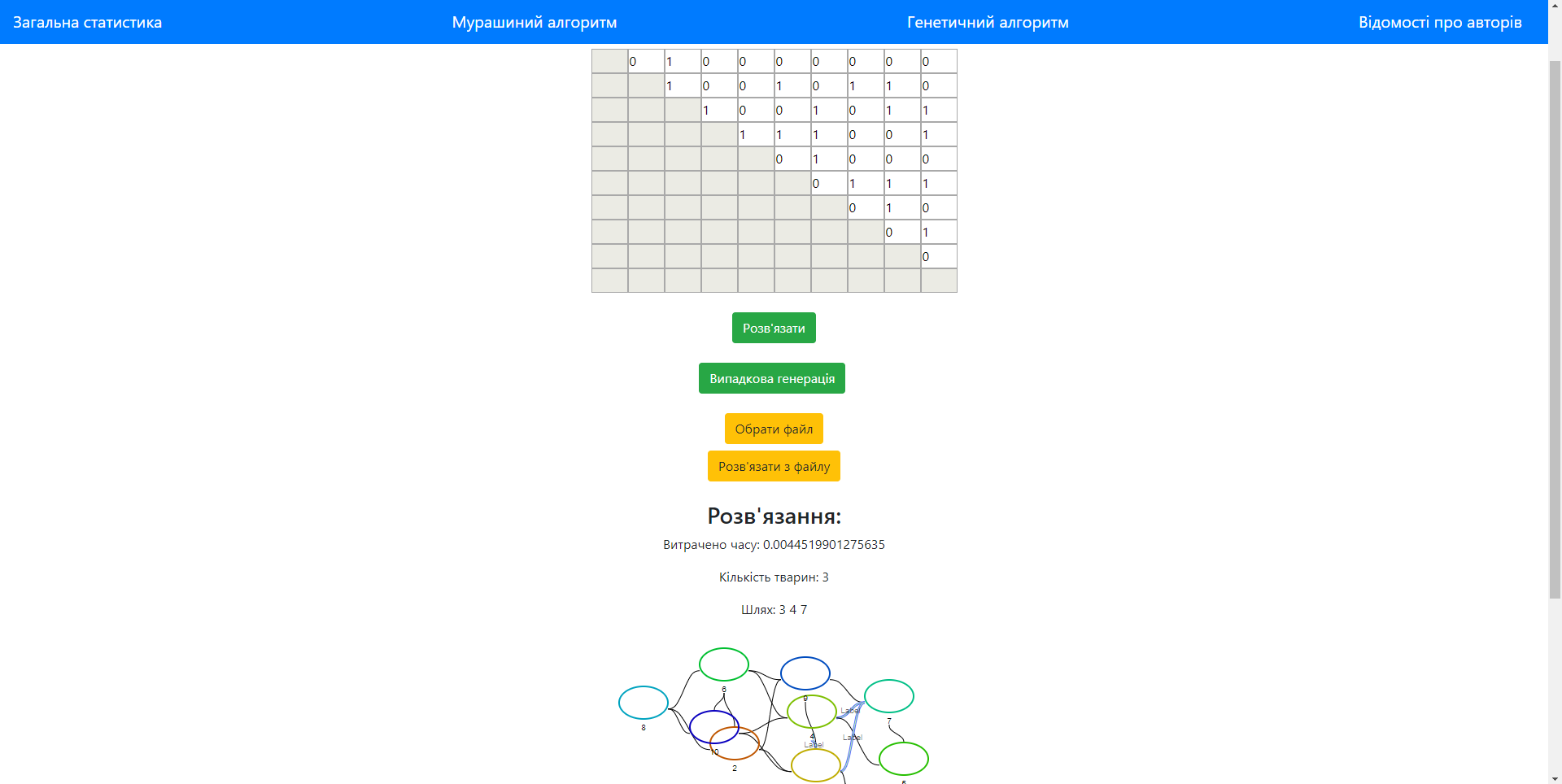


Рисунок 3.6 – Розв’язок задачі

Генетичний алгоритм має ті самі функції, відрізняється лише опис методу (Рисунок 3.7).

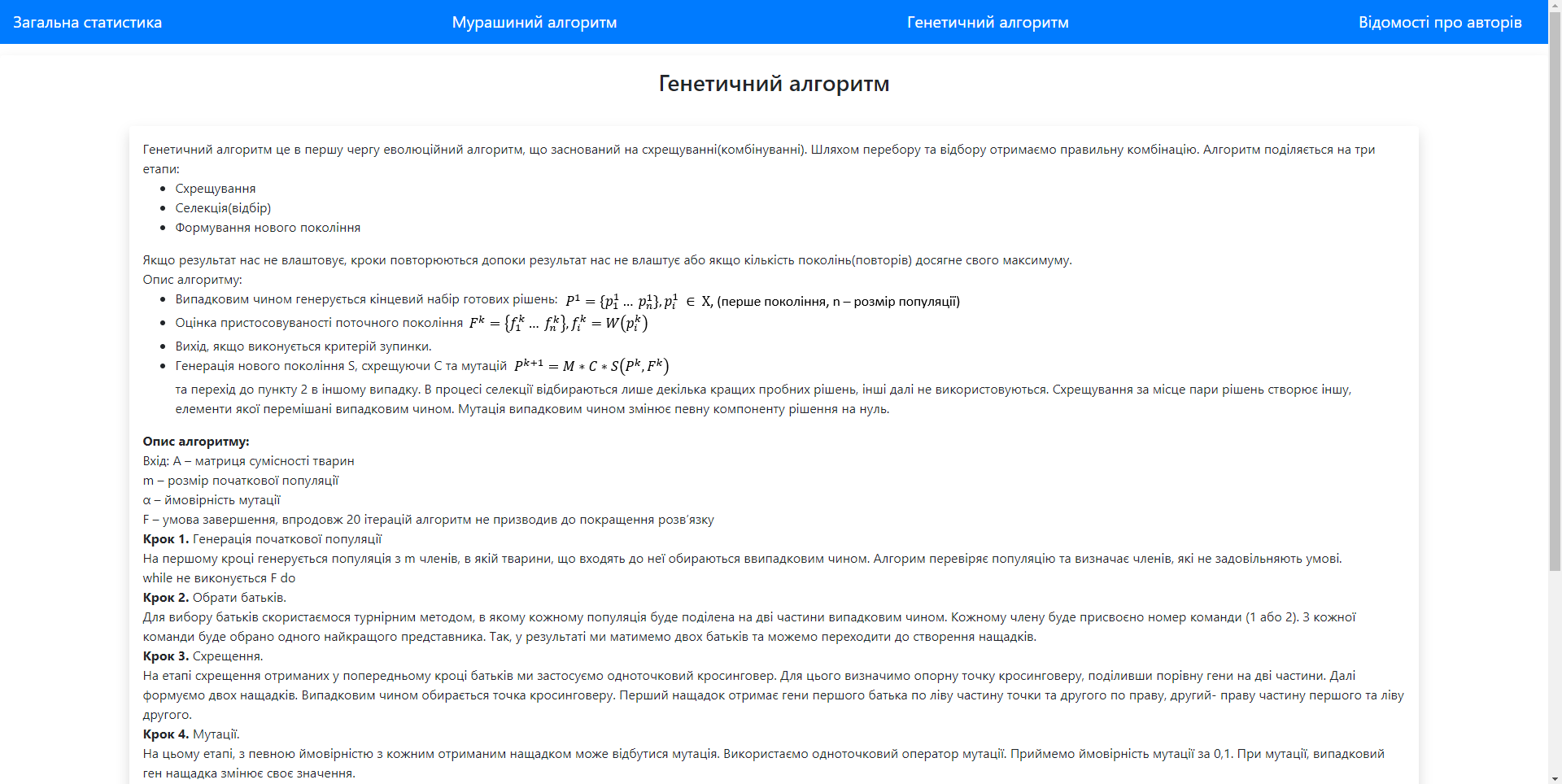


Рисунок 3.7 – Стартова сторінка генетичного алгоритму

Сторінка «відомості про авторів» містить відомості про авторів (Рисунок 3.8).

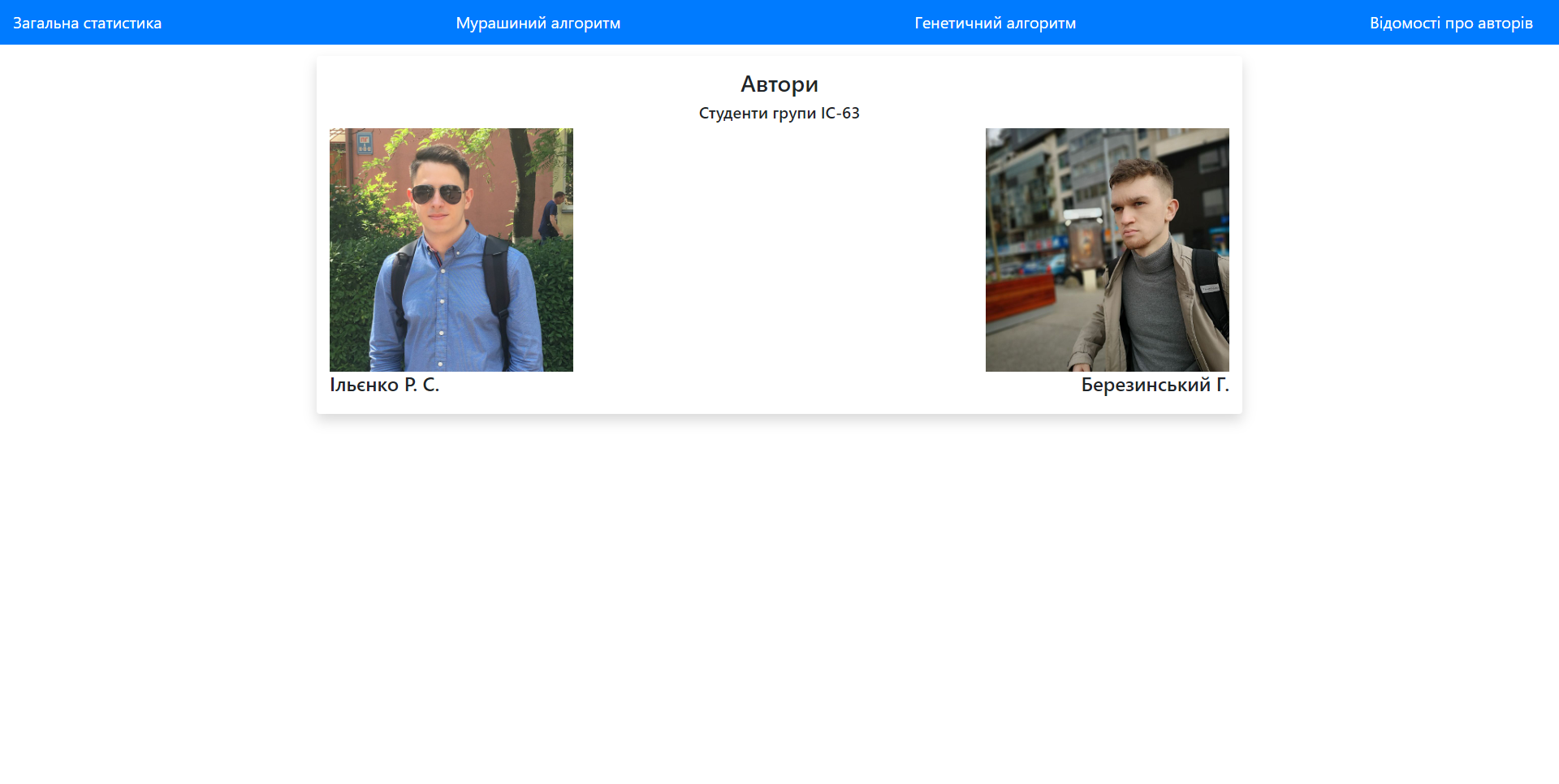


Рисунок 3.8 – Про авторів

## 3.4.2 Методика випробувань

Для випробовування часу роботи кожен з алгоритмів запускався по 10 разів на випадково згенерованих даних розмірністю віл 3 до 10. Алгоритм мурашиних колоній також тестувався на великому наборі даних – від 30 до 35. Генетичний алгоритм обмежився лише тестами невеликої розмірності через надзвичайно високу складність алгоритму, так вже на матриці розмірністю 20 алгоритм працює 16 секунд.

Для випробування точності алгоритмів, кожен запускався на вхідних даних, для яких відоме оптимальне значення ЦФ 1000 разів та вівся підрахунок правильних визначень результатів.

Дослідження алгоритму знаходиться у розділі 4.

# Розділ 4. Дослідження алгоритму

Для проведення експерименту був розроблений програмний продукт. Кожен алгоритм був по 10 разів запущений на випадкових даних розмірності від 3 до 10, тобто було здійснено 70 прогонів алгоритму. На кожній розмірності було визначено середній час роботи алгоритму. За цими даними були побудовані та проаналізовані графіки.

Час роботи генетичного алгоритму в залежності від розмірності представлений у таблиці 1.

Таблиця 1. Час та середній час роботи генетичного алгоритму на 10 запусках в залежності від розміру вхідних даних

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | avg |
| 3 | 0,00079 | 0,00075 | 0,00188 | 0,00086 | 0,00128 | 0,0007 | 0,00073 | 0,00364 | 0,001 | 0,00199 | 0,001362 |
| 4 | 0,00207 | 0,00081 | 0,00185 | 0,00079 | 0,00082 | 0,00194 | 0,00194 | 0,0022 | 0,00089 | 0,00216 | 0,001547 |
| 5 | 0,00092 | 0,00088 | 0,00067 | 0,00082 | 0,00127 | 0,00089 | 0,00112 | 0,00197 | 0,00089 | 0,00082 | 0,001025 |
| 6 | 0,00083 | 0,00097 | 0,00105 | 0,00109 | 0,00084 | 0,00096 | 0,00091 | 0,00086 | 0,00109 | 0,00101 | 0,000961 |
| 7 | 0,00318 | 0,00096 | 0,00252 | 0,00097 | 0,00088 | 0,00364 | 0,00109 | 0,00132 | 0,00227 | 0,00174 | 0,001857 |
| 8 | 0,00214 | 0,00327 | 0,00128 | 0,00163 | 0,0052 | 0,00094 | 0,00169 | 0,0019 | 0,00099 | 0,00508 | 0,002412 |
| 9 | 0,002 | 0,00124 | 0,00632 | 0,00208 | 0,0027 | 0,00149 | 0,00354 | 0,00162 | 0,00207 | 0,00176 | 0,002482 |
| 10 | 0,00289 | 0,00489 | 0,00195 | 0,00473 | 0,00259 | 0,00302 | 0,00442 | 0,00244 | 0,01102 | 0,00641 | 0,004436 |

Рисунок 4.1 - Залежність швидкості від розміру вхідних даних

Графік залежності показує стрімкий зріст часу роботи зі збільшенням розмірності матриці.

Нижче у таблиці 2 представлені час таблиця з часом роботи та графік залежності часу від розмірності мурашиного алгоритму.

Таблиця 2. Час та середній час роботи алгоритму мурашиних колоній на 10 запусках в залежності від розміру вхідних даних

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | avg |
| 3 | 0,002 | 0,00082 | 0,00085 | 0,0018 | 0,00214 | 0,00225 | 0,00086 | 0,0008 | 0,00122 | 0,00079 | 0,001353 |
| 4 | 0,0018 | 0,00247 | 0,0037 | 0,00093 | 0,0015 | 0,00362 | 0,00174 | 0,00363 | 0,0027 | 0,0035 | 0,002559 |
| 5 | 0,00111 | 0,00286 | 0,00243 | 0,00113 | 0,00149 | 0,00163 | 0,00396 | 0,00211 | 0,00629 | 0,0022 | 0,002521 |
| 6 | 0,00234 | 0,0043 | 0,00457 | 0,00289 | 0,00309 | 0,00179 | 0,00337 | 0,00304 | 0,00492 | 0,00345 | 0,003376 |
| 7 | 0,00241 | 0,0028 | 0,00181 | 0,00465 | 0,00298 | 0,00578 | 0,00271 | 0,00234 | 0,00252 | 0,00632 | 0,003432 |
| 8 | 0,00475 | 0,00482 | 0,00309 | 0,00302 | 0,00204 | 0,00227 | 0,00509 | 0,00601 | 0,00306 | 0,00507 | 0,003922 |
| 9 | 0,00402 | 0,00296 | 0,00255 | 0,00379 | 0,00252 | 0,00662 | 0,00247 | 0,00606 | 0,00308 | 0,00318 | 0,003725 |
| 10 | 0,00324 | 0,00406 | 0,00458 | 0,00797 | 0,0049 | 0,00936 | 0,01239 | 0,00388 | 0,00608 | 0,00585 | 0,006231 |

Рисунок 4.2 - Залежність швидкості від розміру вхідних даних

Як видно з графіку, зростання є більш плавним, окрім розмірності 10, де час роботи зростає майже в двічі.

Порівняємо на графіку обидва алгоритми.

Рисунок 4.3 - Порівняльна характеристика швидкості роботи алгоритмів

Де помаранчевим представлений генетичний алгоритм, а синім – мурашиних колоній.

Як видно з двох графіків, швидкість роботи генетичного алгоритму значно вища за швидкість мурашиного, проте це не говорить про перевагу генетичного алгоритму. Перевіримо точність обох алгоритмів, аби прийти до висновку про те, який алгоритм все ж таки краще.

На відомих даних розмірності від 3 до 10, для яких відомі оптимальні розв’язки (відомий оптимальний результат для кожного вхідного набору даних). Кожен алгоритм був запущений 1000 разів та взятий відсоток правильного знаходження оптимальних результатів. Вхідні дані додаються у додатках.

Приведемо графіки точності для генетичного алгоритму та алгоритму мурашиних колоній та їх порівняльний графік.

Рисунок 4.4 - Точність генетичного алгоритму в залежності від розміру вхідних даних.

Рисунок 4.5 - Точність алгоритму мурашиних колоній в залежності від розміру вхідних даних.

Рисунок 4.6 - Порівняльна характеристика точності алгоритмів в залежності від розміру вхідних даних

Де генетичний алгоритм представлений помаранчевим, а мурашиний – синім.

Як видно з дослідження, хоча генетичний алгоритм працює значно швидше, він має надзвичайно низьку точність на вхідних даних розмірності більше 5, в той час, як алгоритм мурашиних колоній має досить непогану точність на усіх вхідних даних. Низька точність в окремих випадках може бути викликана специфічними вхідними даними.

Розглянемо додатково виконання алгоритму на великих наборах даних. Нехай задача задана матрицею з рисунку 4.7.

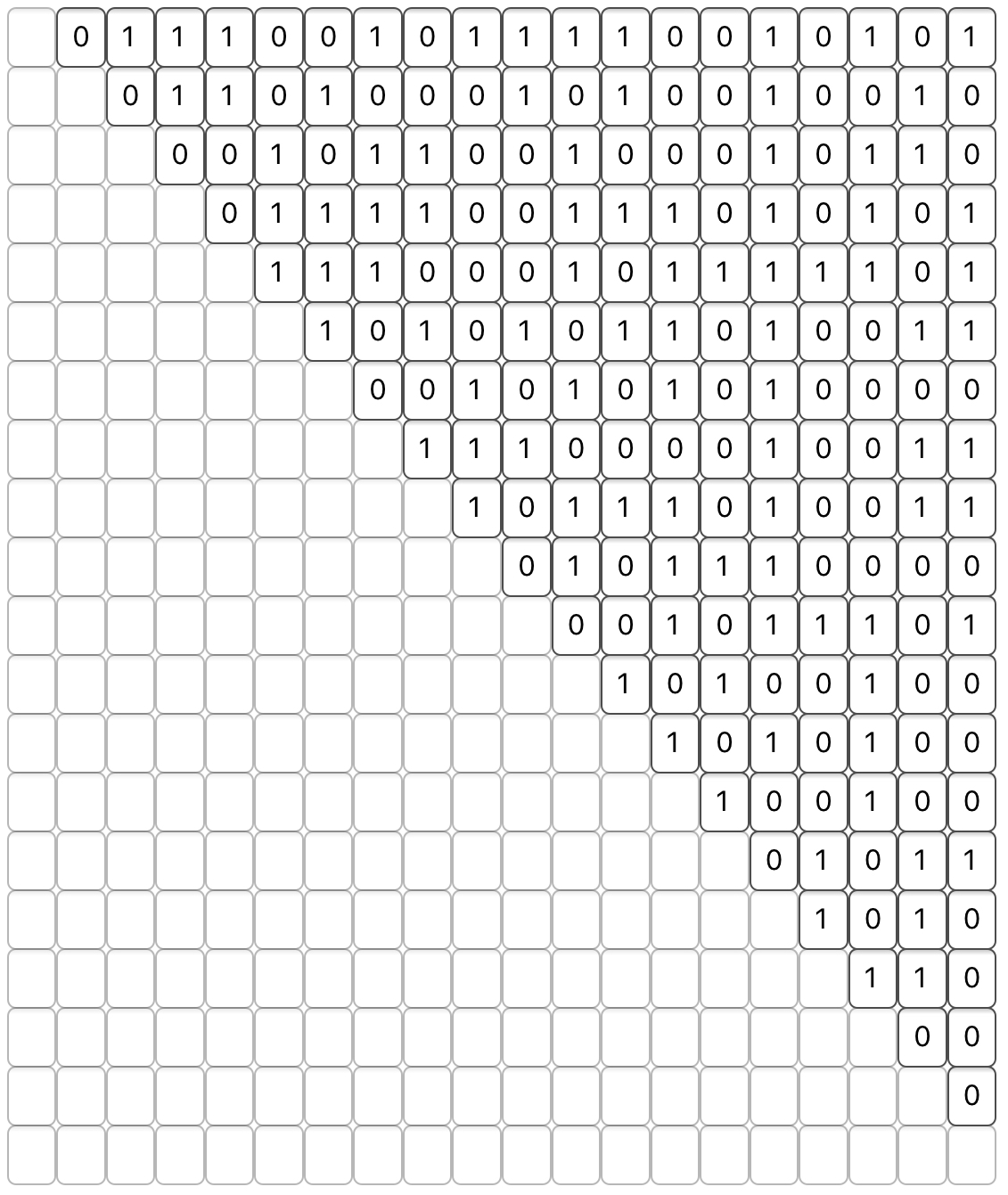


Рисунок 4.7 – Вхідні данні розмірністю 20



Рисунок 4.8 – Результат роботи мурашиного алгоритму



Рисунок 4.9 – Результат роботи генетичного алгоритму

Як висновок можна сказати, що хоч генетичний алгоритм працює швидше, швидкість зростання часу роботи в залежності від вхідних даних значно нижча за мурашиний, він має надзвичайно низьку точність, яка викликана повністю випадковою побудовою розв’язку.

# Додаток А. Список джерел

1. Панченко, Т. В. Генетические алгоритмы [Текст] : учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. — Астрахань : Издательский дом «Астраханский университет», 2007. — 87 [3] с.
2. Муравьиные алгоритмы [Електронний ресурс] - <https://habr.com/ru/post/105302/>

# Додаток Б. Тестові вхідні дані

1. n = 3, z = 2

1, 1, 0

1, 1, 1

0, 1, 1

1. n = 4, z = 3

1, 1, 1, 0

1, 1, 1, 1

1, 1, 1, 0

0, 1, 0, 1

1. n = 5, z = 4

1, 1, 1, 1, 1

1, 1, 0, 1, 1

1, 0, 1, 0, 1

1, 1, 0, 1, 1

1, 1, 1, 1, 1

1. n = 6, z =3

1, 0, 1, 0, 0, 1

0, 1, 0, 0, 1, 1

1, 0, 1, 0, 0, 1

1, 0, 0, 1, 1, 0

0, 1, 0, 1, 1, 0

1, 0, 1, 0, 0, 1

1. n = 7, z = 5

1, 1, 1, 1, 1, 1, 0

1, 1, 1, 1, 0, 1, 0

1, 1, 1, 1, 0, 1, 0

1, 1, 1, 1, 0, 1, 1

1, 0, 0, 0, 1, 0, 0

1, 1, 1, 1, 0, 1, 1

0, 0, 0, 1, 0, 1, 1

1. n = 6, z = 4

1, 1, 1, 0, 0, 1, 0, 0

1, 1, 1, 1, 1, 0, 0, 1

1, 1, 1, 0, 0, 0, 0, 0

0, 1, 0, 1, 1, 1, 0, 1

0, 1, 0, 1, 1, 0, 1, 1

1, 0, 0, 1, 0, 1, 0, 0

0, 0, 0, 0, 1, 0, 1, 1

0, 1, 0, 1, 1, 0, 1, 1

1. n = 9, z = 7

1, 1, 1, 0, 1, 1, 0, 1, 1

1, 1, 1, 0, 1, 1, 0, 1, 1

1, 1, 1, 0, 1, 1, 0, 1, 1

0, 0, 0, 1, 1, 0, 1, 0, 0

1, 1, 1, 1, 1, 1, 0, 1, 1

1, 1, 1, 0, 1, 1, 1, 1, 1

0, 0, 0, 1, 0, 1, 1, 0, 0

1, 1, 1, 0, 1, 1, 0, 1, 1

1, 1, 1, 0, 1, 1, 0, 1, 1

1. n = 10, z = 3

1, 1, 0, 0, 1, 0, 1, 0, 0, 1

1, 1, 1, 0, 0, 0, 0, 0, 0, 0

0, 1, 1, 1, 0, 0, 0, 0, 0, 0

0, 0, 1, 1, 1, 0, 0, 0, 0, 0

1, 0, 0, 1, 1, 1, 1, 0, 0, 0

0, 0, 0, 0, 1, 1, 1, 0, 0, 0

1, 0, 0, 0, 1, 1, 1, 1, 0, 0

0, 0, 0, 0, 0, 0, 1, 1, 1, 0

0, 0, 0, 0, 0, 0, 0, 1, 1, 1

1, 0, 0, 0, 0, 0, 0, 0, 1, 1