**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 5 з дисципліни

«Проектування алгоритмів»

„**Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.2**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-15, Плугатирьов Д.В.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2023

Зміст

[1 Мета лабораторної роботи 3](#_Toc52291748)

[2 Завдання 4](#_Toc52291749)

[3 Виконання 10](#_Toc52291750)

[3.1 Покроковий алгоритм 10](#_Toc52291751)

[3.2 Програмна реалізація алгоритму 10](#_Toc52291752)

[3.2.1 Вихідний код 10](#_Toc52291753)

[3.2.2 Приклади роботи 10](#_Toc52291754)

[3.3 Тестування алгоритму 11](#_Toc52291755)

[Висновок 12](#_Toc52291756)

[Критерії оцінювання 13](#_Toc52291757)

# Мета лабораторної роботи

Мета роботи – вивчити основні підходи розробки метаеврестичних алгоритмів для типових прикладних задач. Опрацювати методологію підбору прийнятних параметрів алгоритму.

# Завдання

Згідно варіанту, формалізувати алгоритм вирішення задачі відповідно загальної методології.

Записати розроблений алгоритм у покроковому вигляді. З достатнім степенем деталізації.

Виконати його програмну реалізацію на будь-якій мові програмування.

Перелік задач наведено у таблиці 2.1.

Перелік алгоритмів і досліджуваних параметрів у таблиці 2.2.

Задача і алгоритм наведені в таблиці 2.3.

Змінюючи параметри алгоритму, визначити кращі вхідні параметри алгоритму. Для цього необхідно:

* обрати критерій зупинки алгоритму (кількість ітерацій або значення ЦФ);
* зафіксувати усі параметри крім одного і змінювати цей параметр, поки не буде досягнуто пікової ефективності;
* після цього параметр фіксується і змінюються інші параметри;
* далі повторюємо процедуру спочатку, з першого зафіксованого параметру;
* зупиняємось коли будуть знайдені оптимальні параметри для даної задачі або встановлена залежність одних параметрів від інших.

Зробити узагальнений висновок в якому обов’язково описати залежність якості розв’язку від вхідних параметрів.

Таблиця 2.1 – Прикладні задачі

|  |  |
| --- | --- |
| **№** | **Задача** |
| 1 | **Задача про рюкзак** (місткість P=500, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 20 (випадкова)). Для заданої множини предметів, кожен з яких має вагу і цінність, визначити яку кількість кожного з предметів слід взяти, так, щоб сумарна вага не перевищувала задану, а сумарна цінність була максимальною.  Задача часто виникає при розподілі ресурсів, коли наявні фінансові обмеження, і вивчається в таких областях, як комбінаторика, інформатика, теорія складності, криптографія, прикладна математика. |
| 2 | **Задача комівояжера** (300 вершин, відстань між вершинами випадкова від 5 до 150) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів.  **Розглядається симетричний, асиметричний та змішаний варіанти.**  В загальному випадку, асиметрична задача комівояжера відрізняється тим, що ребра між вершинами можуть мати різну вагу в залежності від напряму, тобто, задача моделюється орієнтованим графом. Таким чином, окрім ваги ребер графа, слід також зважати і на те, в якому напрямку знаходяться ребра.  У випадку симетричної задачі всі пари ребер між одними й тими самими вершинами мають однакову вагу.  У випадку реальних міст може бути як симетричною, так і асиметричною в залежності від тривалості або довжини маршрутів і напряму руху.  Застосування:   * доставка товарів (в цьому випадку може бути більш доречна постановка транспортної задачі - доставка в кілька магазинів з декількох складів); * доставка води; * моніторинг об'єктів; * поповнення банкоматів готівкою; * збір співробітників для доставки вахтовим методом. |
| 3 | **Розфарбовування графа** (300 вершин, степінь вершини не більше 30, але не менше 2) – називають таке приписування кольорів (або натуральних чисел) його вершинам, що ніякі дві суміжні вершини не набувають однакового кольору. Найменшу можливу кількість кольорів у розфарбуванні називають хроматичне число.  Застосування:   * розкладу для освітніх установ; * розкладу в спорті; * планування зустрічей, зборів, інтерв'ю; * розклади транспорту, в тому числі - авіатранспорту; * розкладу для комунальних служб; |
| 4 | **Задача вершинного покриття** (300 вершин, степінь вершини не більше 30, але не менше 2)**.** Вершинне покриття для неорієнтованого графа G = (V, E) - це множина його вершин S, така, що, у кожного ребра графа хоча б один з кінців входить в вершину з S.  Задача вершинного покриттяполягає в пошуку вершинного покриття найменшого розміру для заданого графа (цей розмір називається числом вершинного покриття графа).  На вході: Граф G = (V, E).  Результат: множина C ⊆ V - найменше вершинне покриття графа G.    Застосування:   * розміщення пунктів обслуговування; * призначення екіпажів на транспорт; * проектування інтегральних схем і конвеєрних ліній. |
| 5 | **Задача про кліку** (300 вершин, степінь вершини не більше 30, але не менше 2)**.** Клікою в неорієнтованому графі називається підмножина вершин, кожні дві з яких з'єднані ребром графа. Іншими словами, це повний підграф первісного графа. Розмір кліки визначається як число вершин в ній.  Задача про кліку існує у двох варіантах: у **задачі розпізнавання** потрібно визначити, чи існує в заданому графі G кліка розміру k, тоді як в **обчислювальному варіанті** потрібно знайти в заданому графі G кліку максимального розміру або всі максимальні кліки (такі, що не можна збільшити).  Застосування:   * біоінформатика; * електротехніка; |
| 6 | **Задача про найкоротший шлях** (300 вершин, відстань між вершинами випадкова від 5 до 150, степінь вершини не більше 10, але не менше 1) - задача пошуку найкоротшого шляху (ланцюга) між двома точками (вершинами) на графі, в якій мінімізується сума ваг ребер, що складають шлях.  Важливість задачі визначається її різними практичними застосуваннями. Наприклад, в GPS-навігаторах здійснюється пошук найкоротшого шляху між точкою відправлення і точкою призначення. Як вершин виступають перехрестя, а дороги є ребрами, які лежать між ними. Якщо сума довжин доріг між перехрестями мінімальна, тоді знайдений шлях найкоротший. |

Таблиця 2.2 – Варіанти алгоритмів і досліджувані параметри

|  |  |
| --- | --- |
| **№** | **Алгоритми і досліджувані параметри** |
| 1 | **Генетичний алгоритм:**   * оператор схрещування (мінімум 3); * мутація (мінімум 2); * оператор локального покращення (мінімум 2). |
| 2 | **Мурашиний алгоритм**:   * α; * β; * ρ; * Lmin; * кількість мурах М і їх типи (елітні, тощо…); * маршрути з однієї чи різних вершин. |
| 3 | **Бджолиний алгоритм:**   * кількість ділянок; * кількість бджіл (фуражирів і розвідників). |

Таблиця 2.3 – Варіанти задач і алгоритмів

|  |  |
| --- | --- |
| **№** | **Задачі і алгоритми** |
| 1 | Задача про рюкзак + Генетичний алгоритм |
| 2 | Задача про рюкзак + Бджолиний алгоритм |
| 3 | Задача комівояжера (асиметрична мережа) + Генетичний алгоритм |
| 4 | Задача комівояжера (симетрична мережа) + Генетичний алгоритм |
| 5 | Задача комівояжера (змішана мережа) + Генетичний алгоритм |
| 6 | Задача комівояжера (асиметрична мережа) + Мурашиний алгоритм |
| 7 | Задача комівояжера (симетрична мережа) + Мурашиний алгоритм |
| 8 | Задача комівояжера (змішана мережа) + Мурашиний алгоритм |
| 9 | Задача вершинного покриття + Генетичний алгоритм |
| 10 | Задача вершинного покриття + Бджолиний алгоритм |
| 11 | Задача комівояжера (асиметрична мережа) + Бджолиний алгоритм |
| 12 | Задача комівояжера (симетрична мережа) + Бджолиний алгоритм |
| 13 | Задача комівояжера (змішана мережа) + Бджолиний алгоритм |
| 14 | Розфарбовування графа + Генетичний алгоритм |
| 15 | Розфарбовування графа + Бджолиний алгоритм |
| 16 | Задача про кліку (задача розпізнавання) + Генетичний алгоритм |
| 17 | Задача про кліку (задача розпізнавання) + Бджолиний алгоритм |
| 18 | Задача про кліку (обчислювальна задача) + Генетичний алгоритм |
| 19 | Задача про кліку (обчислювальна задача) + Бджолиний алгоритм |
| 20 | Задача про найкоротший шлях + Генетичний алгоритм |
| 21 | Задача про найкоротший шлях + Мурашиний алгоритм |
| 22 | Задача про найкоротший шлях + Бджолиний алгоритм |
| 23 | Задача про рюкзак + Генетичний алгоритм |
| 24 | Задача про рюкзак + Бджолиний алгоритм |
| 25 | Задача комівояжера (асиметрична мережа) + Генетичний алгоритм |
| 26 | Задача комівояжера (симетрична мережа) + Генетичний алгоритм |
| 27 | Задача комівояжера (змішана мережа) + Генетичний алгоритм |
| 28 | Задача комівояжера (асиметрична мережа) + Мурашиний алгоритм |
| 29 | Задача комівояжера (симетрична мережа) + Мурашиний алгоритм |
| 30 | Задача комівояжера (змішана мережа) + Мурашиний алгоритм |

# Виконання

## Покроковий алгоритм

### Основний алгоритм

|  |  |
| --- | --- |
| **Змінна** | **Призначення** |
| iterationsCount | К-сть ітерацій алгоритму |
| i | Лічильник циклу |
| offspring | Нащадок |
| maximalProbabilityLevel | Максимально доступне значення ймовірності мутації |
| mutationProbability | Випадково згенерована ймовірність мутації |

1. ПОЧАТОК
2. Згенерувати хромосоми
3. Визначити maximalProbabilityLevel
4. ЦИКЛ ПОКИ i < iterationsCount
5. Провести схрещення
6. ЯКЩО offspring == null ТО
7. continue
8. Випадково згенерувати mutationProbability
9. ЯКЩО mutationProbability <= maximalProbabilityLevel ТО
10. Застосувати мутацію до offspring
11. Застосувати оператор покращення до offspring
12. Додати offspring до популяції
13. Видалити хромосому із найбільшою довжиною між початковою та кінцевою вершинами
14. i = i + 1
15. КІНЕЦЬ ЦИКЛУ
16. Повернути найкоротший шлях

### Алгоритм створення хромосоми

|  |  |
| --- | --- |
| Змінна | Призначення |
| chromosomeToAdd | Хромосома, котра в процесі створення |

* + - 1. ПОЧАТОК
      2. Створити пусту хромосому chromosomeToAdd
      3. Додати до chromosomeToAdd початкову вершину
      4. ПОЧАТОК ЦИКЛУ
      5. Отримати всі дуги, протилежні вершини яких можна з’єднати без повторів із останньою доданою вершиною chromosomeToAdd
      6. ЯКЩО к-сть знайдених дуг нульова ТО
      7. Видалити всі вершини із chromosomeToAdd, окрім першої
      8. continue
      9. Отримати випадковий індекс одної зі знайдених дуг
      10. Отримати число вершини із дуги за індексом
      11. Додати знайдений номер вершини до chromosomeToAdd
      12. ЯКЩО існує хромосома із такими самими номерами вершин ТО
      13. Видалити всі номери вершин у chromosomeToAdd, окрім початкового
      14. ПОКИ останній номер chromosomeToAdd не дорівнює кінцевому номеру
      15. Додати chromosomeToAdd до популяції

**Вибір батьків**

Під час вибору батьків я надав перевагу пропорційній селекції на противагу вибору кращого в популяції та випадкового батьків. Це збільшує шанси на те, що знайдеться більш оптимальний шлях, адже найкращий у популяції буде лише псувати збагачення нащадків «свіжими» генами.

**Оператор схрещування**

Мною було використано одноточковий та рівномірний оператори. Судячи із висновків роботи алгоритму, одноточковий краще себе показує на великих графах, а рівномірний – на малих. Оскільки особливістю рівномірного оператору схрещування є його нестабільний випадковий переніс генів то з однієї хромосоми, то з іншої до нащадка без урахування можливості генів мати спільні дуги, через що багато із них не виживають.

Одноточковий же оператор просто схрещує хромосоми за спільною вершиною, якщо така існує з можливістю як набуття, так і втрати генів. Із часом, останнє переважає, що змушує хромосоми приймати однаковий набір генів на великих ітераціях.

**Оператор мутації**

В якості оператору мутації мною було використано вставку випадкового гена у випадкове місце хромосоми. Це дозволило мені знаходити нові шляхи більш ефективно.

**Локальне покращення**

В якості локального покращення я у випадкове місце хромосоми вставляв ген, який, на відміну від усіх інших, маючих спільні дуги із його сусідами, зменшував довжину хромосоми найбільше.

## Програмна реалізація алгоритму

### Вихідний код

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

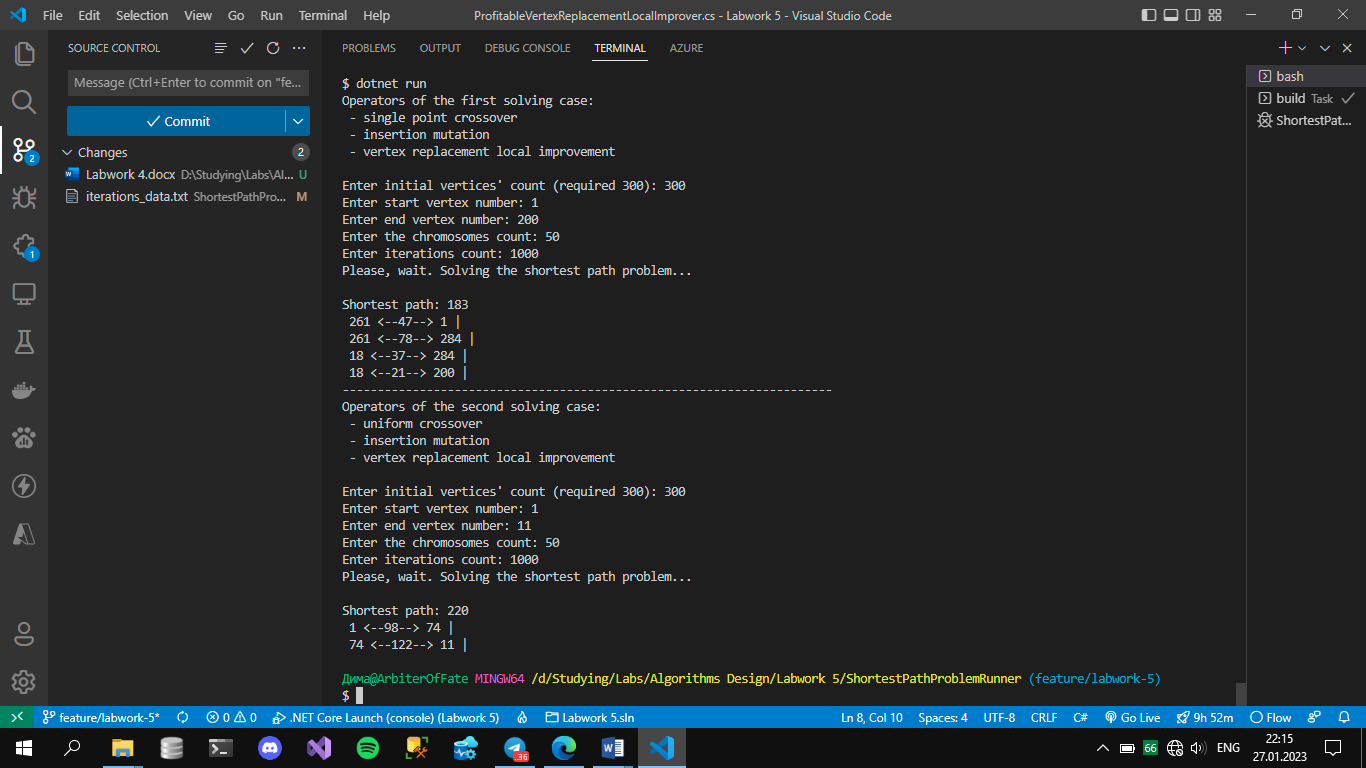


Рисунок 3.1 – результат роботи програми для одноточкового та рівномірного операторів схрещування

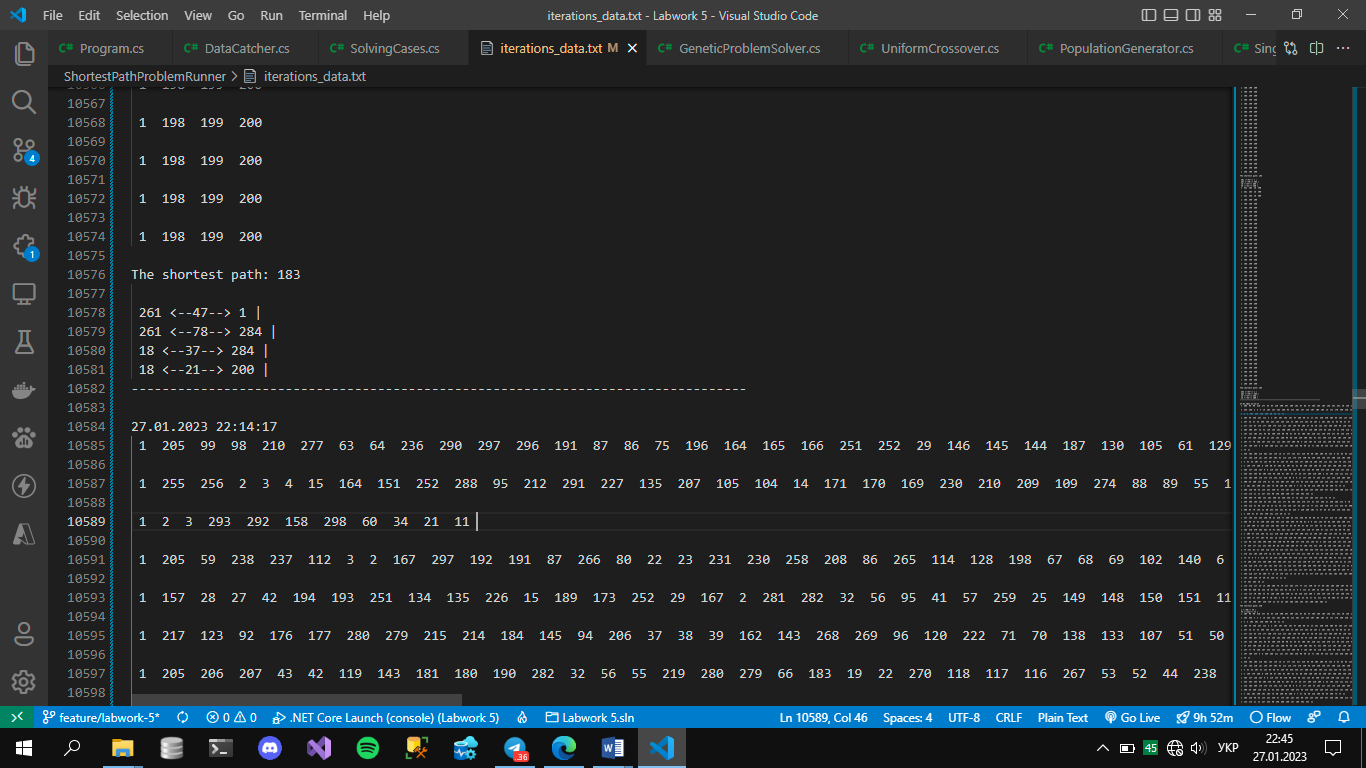


Рисунок 3.2 – запис до текстового файлу значень хромосом та найкоротшого шляху разом із його представленням у вигляді сукупності дуг кожні 10 ітерацій програми

## Тестування алгоритму

Перед переходом до суті, доречно буде описати існуючі обмеження, які визначаються умовою задачі. Тобто, в результаті я маю граф на 300 вершин, степені яких коливаються від 1 до 10 включно, а відстані між ними від 5 до 150 включно. Також, окрім цього, є ще параметри, які впливають на поведінку алгоритму.

### Дослідження параметру мутації

Мутація в генетичному алгоритмі несе за собою невизначеність та сприяє захопленню нових вершин графу в околицях поточних рішень. Вона сприяє розростанню популяції та її бажанню не сидіти на місці.

Таблиця 3.1 – значення цільової функції зі зміною параметра мутації

|  |  |  |
| --- | --- | --- |
| Ймовірність мутації | Номер дослідження | Значення цільової функції |
| 0 | 1 | 164 |
| 2 | 208 |
| 3 | 145 |
| 4 | 190 |
| 5 | 139 |
| Середнє | 169 |
| 0,2 | 1 | 179 |
| 2 | 222 |
| 3 | 217 |
| 4 | 220 |
| 5 | 175 |
| Середнє | 202 |

Продовження таблиці 3.1

|  |  |  |
| --- | --- | --- |
| Ймовірність мутації | Номер дослідження | Значення цільової функції |
| 0,4 | 1 | 280 |
| 2 | 373 |
| 3 | 234 |
| 4 | 484 |
| 5 | 177 |
| Середнє | 309 |
| 0,6 | 1 | 227 |
| 2 | 141 |
| 3 | 204 |
| 4 | 158 |
| 5 | 161 |
| Середнє | 178 |
| 0,8 | 1 | 176 |
| 2 | 183 |
| 3 | 180 |
| 4 | 315 |
| 5 | 76 |
| Середнє | 186 |
| 1 | 1 | 143 |
| 2 | 185 |
| 3 | 227 |
| 4 | 234 |
| 5 | 190 |
| Середнє | 195 |

Рисунок 3.3 – залежність значень цільової функції від імовірності мутації

Висновок

В рамках даної лабораторної роботи я застосував оптимальне рішення проблеми найкоротшого шляху на практиці за допомогою генетичного алгоритму.

Це сталося завдяки тому, що я підбирав прийнятні параметри для вирішення конкретної проблеми. Я створив оператори схрещування, мутації та локального покращення, кожен вид яких діяв по-своєму, направлений на конкретну область застосування алгоритму.

Наприклад, мутація, котра спрацьовує над нащадком в залежності від завчасно нарахованої ймовірності, ефективна не завжди. Як можна помітити, я створив графік, котрий відображає залежність між значеннями цільової функції та ймовірністю мутації. Проте, її застосування стає доцільним лише за малих або великих шансів спрацювання.

Під час виконання завдань, мною було прийняте рішення записувати дані, отримані під час роботи програми, до файлу, котрий розташовується у проєкті, що запускає програму.

Критерії оцінювання

При здачі лабораторної роботи до 11.12.2022 включно максимальний бал дорівнює – 5. Після 11.12.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* покроковий алгоритм – 15%;
* програмна реалізація алгоритму – 50%;
* тестування алгоритму– 30%;
* висновок – 5%.