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”
Варіант 25

Виконав(ла) *ІП-15, Тонконог В.В.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Ахаладзе І.Е.* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 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.

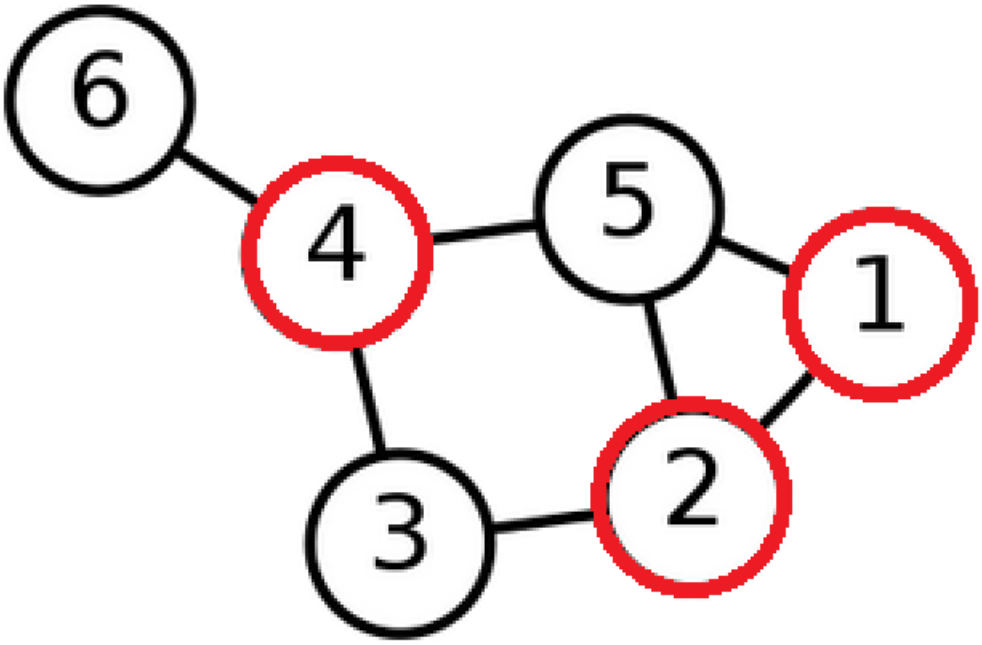
Зробити узагальнений висновок в якому обов’язково описати залежність якості розв’язку від вхідних параметрів.

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 покриття полягає 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 M 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

3 ВИКОНАННЯ

Варіант 25

3.1 Покроковий алгоритм

3.1.1 Відбір двох батьків серед популяції

3.1.2 Схрещування обраних батьків

3.1.3 Мутація отриманого нащадка

3.1.4 Локальна оптимізація отриманого нащадка

3.1.5 Якщо, отриманий нащадок краща за найгіршу особину в популяції, то він замінює її.

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

Вихідний код розміщено у репозиторії: https://github.com/Triplix1/PA

3.2.2 Приклади роботи

На рисунку 3.1 показано відстані між містами, на рисунках 3.2 і 3.3 показані приклади роботи програми.

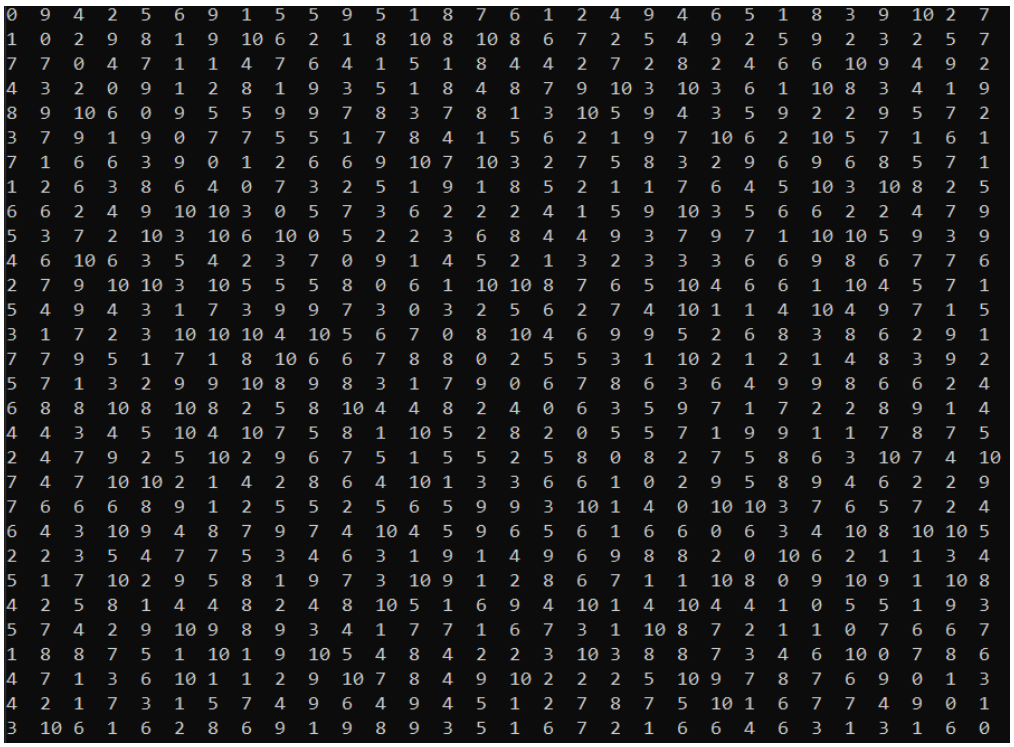


Рисунок 3.1 – Вхідна таблиця відстані між містами

```
0 28 2 5 29 15 12 4 20 6 16 22 26 3 8 17 11 24 27 7 25 9 23 14 19 13 1 10 21 18 0
48
```

Рисунок 3.2 – приклад роботи на 1000 ітерацій

```
0 17 25 24 2 19 13 1 10 15 12 28 5 23 27 6 8 26 7 14 4 21 18 20 16 22 11 29 9 3 0
47
```

Рисунок 3.3 – приклад роботи на 10000 ітерацій

3.3 Тестування алгоритму

Для тестування алгоритму було створені наступні варіації параметрів:

1. Варіанти відбору батьків:
   1.1. Турнірна селекція – де обираються кілька особин з популяції, з них обирається один найкращий, такі операції повторюються кілька разів доки, попередньо задану розробником. Два кращих серед відібраних кращих повертаються як батьки.
   1.2. Пропорційна селекція – обирається краща особина, і якась випадкова (шанс відбору залежить від значення її функції).
2. Види схрещування:
   2.1. Точкове – обирається точка і до неї копіюється один з батьків, після неї в порядкі черги додаються незайняті вершини другого предку, які стоять після точки схрещування, і якщо цих вершин не достатньо то додаються незайняті вершини першого предка у порядку черги.
   2.2. Порядкове – починаючи з N/4 позиції по 3N/4 копіюється один з батьків, потім незаповнені елементи підряд заповнюються елементмми іншого нащадка в порядку черги.
   2.3. Частинно заповнений - починаючи з N/4 позиції по 3N/4 копіюється один з батьків, потім вставляються відповідні незаповнені елементи відповідними елементами іншого батька, якщо такого елемента ще немає у згенерованому нащадку. Всі пість елементи заповнюються елементами першого батька в порядку черги.
3. Види мутацій
   3.1. Випадкова заміна – обираються два випадкових міста, і міняються місцями.
   3.2. Реверсивна заміна – обираються дві випадкові точки і міста між ними розвертаються.

4. Види локального покращення:
    4.1. Покращення з чотирьох – обирається чотири міста починаючи з першого, міняються місцями два середніх міста (друге та третє), обраховується отимана довжина шляху, якщо ця довжина краща за попередню, то міста на цих місцях і залишаються якщо ні, то вони повертаються в початкове положення, потім обирається наступні чотири міста, і з ними проводяться ці ж самі маніпуляції. Таких четвірок буде N-3.
    4.2. Локальне покращення обертанням – обираються дві випадкові точки, міста між ними розветаються, якщо така операція покращує наш шлях, то ці зміни залишаються, якщо ні, то повертаються до пепереднього розташування. Така оперція проводиться якусь певну сталу кількість разів (в моєму випадку 5).

Для випробування бралися сталими наступні величчини: кількість ітерацій – 1000, кількість міст – 30, матриця відстаней – рисунок 3.1, початкова популяція – 30 особин, шанс мутації – 0,05, точка(для схрещування №1) – 15.

Результатом бралося середнє значення з трьох випробувань.

№	Відбір батьків	Схрещування	Мутація	Локальна оптимізація	Результат
1	1	1	1	1	64
2	1	1	1	2	81
3	1	1	2	1	54
4	1	1	2	2	74
5	1	2	1	1	61
6	1	2	1	2	80
7	1	2	2	1	53
8	1	2	2	2	77
9	1	3	1	1	64
10	1	3	1	2	87

№	Відбір батьків	Схрещування	Мутація	Локальна оптимізація	Результат
11	1	3	2	1	51
12	1	3	2	2	77
13	2	1	1	1	55
14	2	1	1	2	83
15	2	1	2	1	53
16	2	1	2	2	74
17	2	2	1	1	63
18	2	2	1	2	78
19	2	2	2	1	59
20	2	2	2	2	78
21	2	3	1	1	60
22	2	3	1	2	81
23	2	3	2	1	54
24	2	3	2	2	68

Таблиця 3.1 – Резульати випробувань в залежності від параметрів

За результатами випробування можемо чітко сказати, що в середньому випадку дещо краще пропрційна селекція (але найкращого результату було досягнуто за турнірної селекції), оператори схрещування майже не вдрізняються один від одного (але, все ж таки, кращий результат був досягнутий саме за третього оператора), Реверсивна мутація точно краща ніж випадкова заміна, і значно краще працює локальне покращення з чотирьох.

В результаті було досягнуто кращого результату – 51. Це сталося при наступних параметрах: спосіб відбору батьків - турнірна селекція, тип схрещування – частинно заповнене , вид мутації – реверсивна, локальна оптимізація – з чотирьох.

ВИСНОВОК

В рамках даної лабораторної роботи реалізовано генетичний алгоритм для оптимізації роз’вязку задачі комівояжера, протестовано його роботу з різними видами кросовера, вибору батьків, мутації, локальної оптимізації. В результаті було виявлено, що найкраще працює поєднання наступних параметрів: спосіб відбору батьків - турнірна селекція, тип схрещування – частинно заповнене , вид мутації – реверсивна, локальна оптимізація – з чотирьох. Це дещо суперечить середнім показникам, якщо розглядати кожен параметр окремо, адже в загальному випадку краще прцює пропорційна селекція аніж турнірна.

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%.