

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)

ІП-13 Замковий Д.В.

(шифр, прізвище, ім'я, по батькові)

Перевірив

Головченко М.Н.

(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
2	ЗАВДАННЯ	4
3	ВИКОНАННЯ.....	11
3.1	ПОКРОКОВИЙ АЛГОРИТМ	11
3.2	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ	11
3.2.1	<i>Вихідний код.....</i>	<i>11</i>
3.2.2	<i>Приклади роботи</i>	<i>14</i>
3.3	ТЕСТУВАННЯ АЛГОРИТМУ	ПОМИЛКА! ЗАКЛАДКУ НЕ ВИЗНАЧЕНО.
	ВИСНОВОК	15
	КРИТЕРІЇ ОЦІНЮВАННЯ	16

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 ступенем 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.

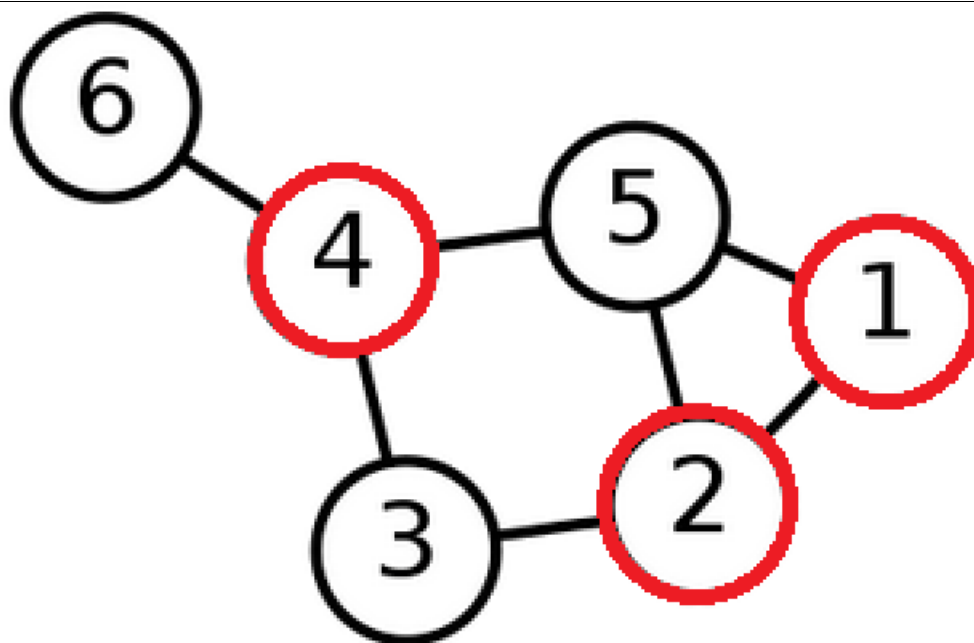
Зробити узагальнений висновок в якому обов'язково описати залежність якості розв'язку від вхідних параметрів.

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

	<p>сумарна вага не перевищувала задану, а сумарна цінність була максимальною.</p> <p>Задача часто виникає при розподілі ресурсів, коли наявні фінансові обмеження, і вивчається в таких областях, як комбінаторика, інформатика, теорія складності, криптографія, прикладна математика.</p>
2	<p>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.</p> <p>Розглядається симетричний, асиметричний та змішаний варіанти.</p> <p>В загальному випадку, асиметрична задача комівояжера відрізняється тим, що ребра між вершинами можуть мати різну вагу в залежності від напрямку, тобто, задача моделюється орієнтованим графом. Таким чином, окрім ваги ребер графа, слід також зважати і на те, в якому напрямку знаходяться ребра.</p> <p>У випадку симетричної задачі всі пари ребер між одними й тими самими вершинами мають однакову вагу.</p> <p>У випадку реальних міст може бути як симетричною, так і асиметричною в залежності від тривалості або довжини маршрутів і напрямку руху.</p> <p>Застосування:</p> <ul style="list-style-type: none"> — доставка товарів (в цьому випадку може бути більш доречна постановка транспортної задачі - доставка в кілька магазинів з декількох складів); — доставка води;

	<ul style="list-style-type: none"> – моніторинг об'єктів; – поповнення банкоматів готівкою; – збір співробітників для доставки вахтовим методом.
3	<p>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.</p> <p>Застосування:</p> <ul style="list-style-type: none"> – розкладу для освітніх установ; – розкладу в спорті; – планування зустрічей, зборів, інтерв'ю; – розклади транспорту, в тому числі - авіатранспорту; – розкладу для комунальних служб;
4	<p>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.</p> <p>Задача вершинного покриття полягає в пошуку вершинного покриття найменшого розміру для заданого графа (цей розмір називається числом вершинного покриття графа).</p> <p>На вході: Граф $G = (V, E)$.</p> <p>Результат: множина $C \subseteq V$ - найменше вершинне покриття графа G.</p>



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

	<p>не менше 1) - задача пошуку найкоротшого шляху (ланцюга) між двома точками (вершинами) на графі, в якій мінімізується сума ваг ребер, що складають шлях.</p> <p>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.</p>
--	--

Таблиця 2.2 – Варіанти алгоритмів і досліджувані параметри

№	Алгоритми і досліджувані параметри
1	<p>Генетичний алгоритм:</p> <ul style="list-style-type: none"> - оператор схрещування (мінімум 3); - мутація (мінімум 2); - оператор локального покращення (мінімум 2).
2	<p>Мурашиний алгоритм:</p> <ul style="list-style-type: none"> – α; – β; – ρ; – L_{min}; – кількість мурах M і їх типи (елітні, тощо...); – маршрути з однієї чи різних вершин.
3	<p>Бджолиний алгоритм:</p> <ul style="list-style-type: none"> – 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

3.1 Покроковий алгоритм

1. Створити граф
2. Знайти рішення жадібним алгоритмом
3. Поки є бджоли-розвідники
 - a. Перейти на випадково вибрану ділянку
 - b. Поки є бджоли-працівники
 - i. Знайти квітку з найбільшим степенем
 - ii. Додати знайдену квітку до покриття
 - iii. Вилучити сусідів з покриття
 - c. Якщо ЦФ покращилось, то оновити граф
 - d. Інакше – залишити без змін

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace ConsoleApp5
{
    [Serializable]
    class Node
    {
        public int num;
        public bool captur;
        public List<int> nbs;

        public Node()
        {
            nbs = new List<int>();
            captur = false;
            num = -1;
        }

        public bool connect(Node nb, int max_c)
        {
            if (!(nb.nbs.Count == max_c) && !this.nbs.Contains(nb.num))
            {
                this.nbs.Add(nb.num);
                nb.nbs.Add(this.num);
                return true;
            }
            return false;
        }
    }
}
```

```

}

class Program {
    const int AMOUNT_OF_NODES = 300;
    const int MAX_COUNT = 30;
    const int MIN_COUNT = 2;
    const int AGENT = 100;
    const int WORKERS = 300;

    static void Main(string[] args)
    {
        var G = create_graph(AMOUNT_OF_NODES, MAX_COUNT, MIN_COUNT);
        greedy_capture(ref G);
        Console.WriteLine("Before: " + rate_solution(G));
        bee_algorithm(ref G);
        Console.WriteLine("After: " + rate_solution(G));
    }

    static void bee_algorithm(ref Node[] G)
    {
        for (int i = 0; i < AGENT; ++i)
        {
            var rand = new Random();
            int rand_index = rand.Next(0, AMOUNT_OF_NODES);
            G = near_search(G, rand_index);
        }
    }

    static Node[] near_search(Node[] G, int index)
    {
        Node node = G[index];
        Node[] back_copy = deep_copy(G);
        for (int i = 0; i < WORKERS; ++i)
        {
            int max_degree = node.nbs.Count;
            int max_index = index;
            foreach (var neigh in node.nbs)
            {
                if (G[neigh].nbs.Count > max_degree)
                {
                    max_degree = G[neigh].nbs.Count;
                    max_index = neigh;
                }
            }
            G[max_index].captur = true;
            for (int j = 0; j < G[max_index].nbs.Count; ++j)
            {
                G[G[max_index].nbs[j]].captur = false;
            }
            greedy_capture(ref G);
        }
        if (rate_solution(G) > rate_solution(back_copy)) return back_copy;
        else return G;
    }

    public static T deep_copy<T>(T item)
    {
        BinaryFormatter formatter = new BinaryFormatter();
        MemoryStream stream = new MemoryStream();
        formatter.Serialize(stream, item);
        stream.Seek(0, SeekOrigin.Begin);
        T result = (T)formatter.Deserialize(stream);
    }
}

```

```

        stream.Close();
        return result;
    }

    static int rate_solution(Node[] graph)
    {
        int res = 0;
        foreach (var node in graph)
        {
            if (node.captur) res++;
        }
        return res;
    }

    static bool is_covered(Node[] nodes, Node node)
    {
        if (node.captur) return true;
        for (int i = 0; i < node.nbs.Count; ++i)
        {
            if (nodes[node.nbs[i]].captur) return true;
        }
        return false;
    }

    static Node[] create_graph(int amount, int max_connections, int min_connections)
    {
        Node[] nodes = new Node[amount];
        for (int i = 0; i < amount; ++i)
        {
            nodes[i] = new Node();
            nodes[i].num = i;
        }
        Random rand = new Random();

        for (int i = 0; i < nodes.Length; ++i)
        {
            int num_connections = rand.Next(min_connections, max_connections);

            for (int j = nodes[i].nbs.Count; j < num_connections; ++j)
            {
                int randIndex = rand.Next(0, nodes.Length);
                if (randIndex == i)
                {
                    if (randIndex == nodes.Length - 1)
                    {
                        randIndex = 0;
                    }
                    else
                    {
                        randIndex++;
                    }
                }
                if (!nodes[i].connect(nodes[randIndex], max_connections)) j--;
            }
        }
        return nodes;
    }

    static void greedy_capture(ref Node[] nodes)
    {
        foreach (var node in nodes)
        {
            if (!is_covered(nodes, node))
            {

```

```

        node.captur = true;
    }
}
}
}
}

```

3.2.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

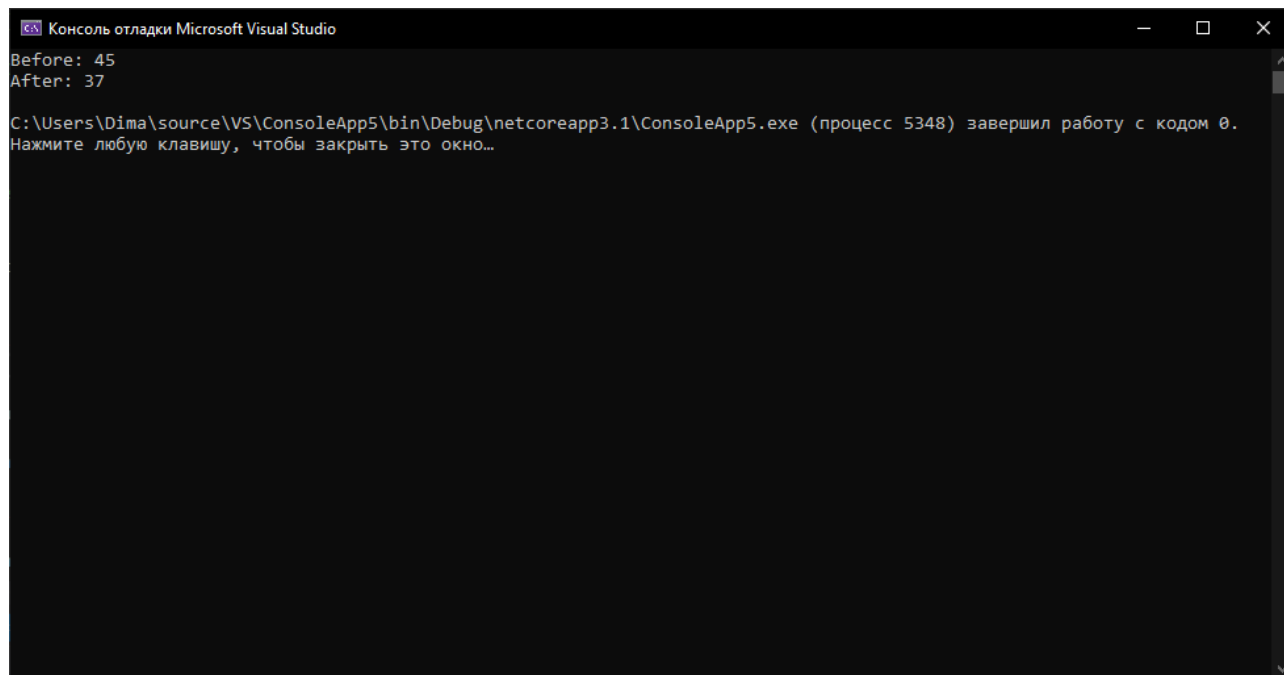


Рисунок 3.1 – пример работы программы

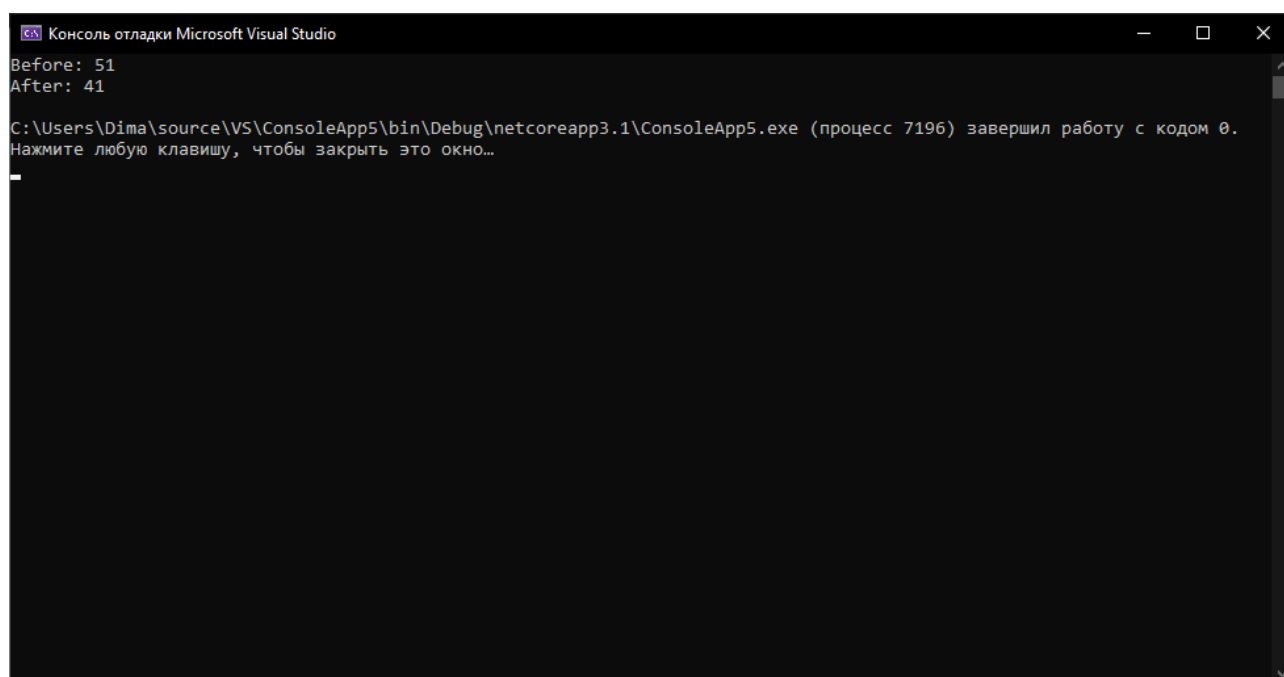


Рисунок 3.2 – пример работы программы

Висновок

В рамках даної лабораторної роботи було досліджено бджолиний алгоритм і набуто практичних знаходження задовільного розв'язку NP-складних задач за допомогою цього алгоритму на прикладі задачі вершинного покриття графу.

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%.