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 № 4 з дисципліни
«Проектування алгоритмів»

„Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1”

Виконав(ла) *ІП-15 Шабанов Метін Шаміль огли* _____
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

1 Мета лабораторної роботи

Мета роботи – вивчити основні підходи формалізації метаевристичних алгоритмів і вирішення типових задач з їхньою допомогою.

2 Завдання

Згідно варіанту, розробити алгоритм вирішення задачі і виконати його програмну реалізацію на будь-якій мові програмування.

Задача, алгоритм і його параметри наведені в таблиці 2.1.

Зафіксувати якість отриманого розв'язку (значення цільової функції) після кожних 20 ітерацій до 1000 і побудувати графік залежності якості розв'язку від числа ітерацій.

Зробити узагальнений висновок.

Таблиця 2.1 – Варіанти алгоритмів

27	Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники).

3 Виконання

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
def scout_bee_work(self):
    if len(self.available_vertexes) > 2:
        best_nodes = random.sample(self.available_vertexes, 2)
        if self.graph.adjacency_matrix[best_nodes[0]].count(1) >
self.graph.adjacency_matrix[best_nodes[1]].count(1):
            return best_nodes[0]
        else:
            return best_nodes[1]
    else:
        return self.available_vertexes[0]

def onlooker_bee_work(self, best_node):
    self.__color_node(best_node)
    self.__color_neighbors(best_node)
    self.available_vertexes.remove(best_node)

def __color_node(self, best_node):
    color_iter = 0
    self.graph.colors[best_node] = [self.available_colors[color_iter], 0]
    while self.check_same_color(self.available_colors[color_iter], best_node):
        color_iter += 1
        self.graph.colors[best_node] = [self.available_colors[color_iter], 0]
    if not self.available_colors[color_iter] in self.used_colors:
        self.used_colors.append(self.available_colors[color_iter])

def __color_neighbors(self, best_node):
    color_iter = 0
    for i in range(len(self.graph.adjacency_matrix[best_node])):
        if self.graph.adjacency_matrix[best_node][i] == 1 and
self.graph.colors[i][1] != 0:
            self.graph.colors[i][0] = self.available_colors[color_iter]
            while self.check_same_color(self.graph.colors[i][0], i):
                color_iter += 1
                self.graph.colors[i][0] = self.available_colors[color_iter]
            if not self.available_colors[color_iter] in self.used_colors:
                self.used_colors.append(self.available_colors[color_iter])
        color_iter = 0

def check_same_color(self, color, node):
    for i in range(len(self.graph.adjacency_matrix[node])):
        if self.graph.adjacency_matrix[node][i] == 1 and self.graph.colors[i][0]
== color:
            return True
    return False
```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

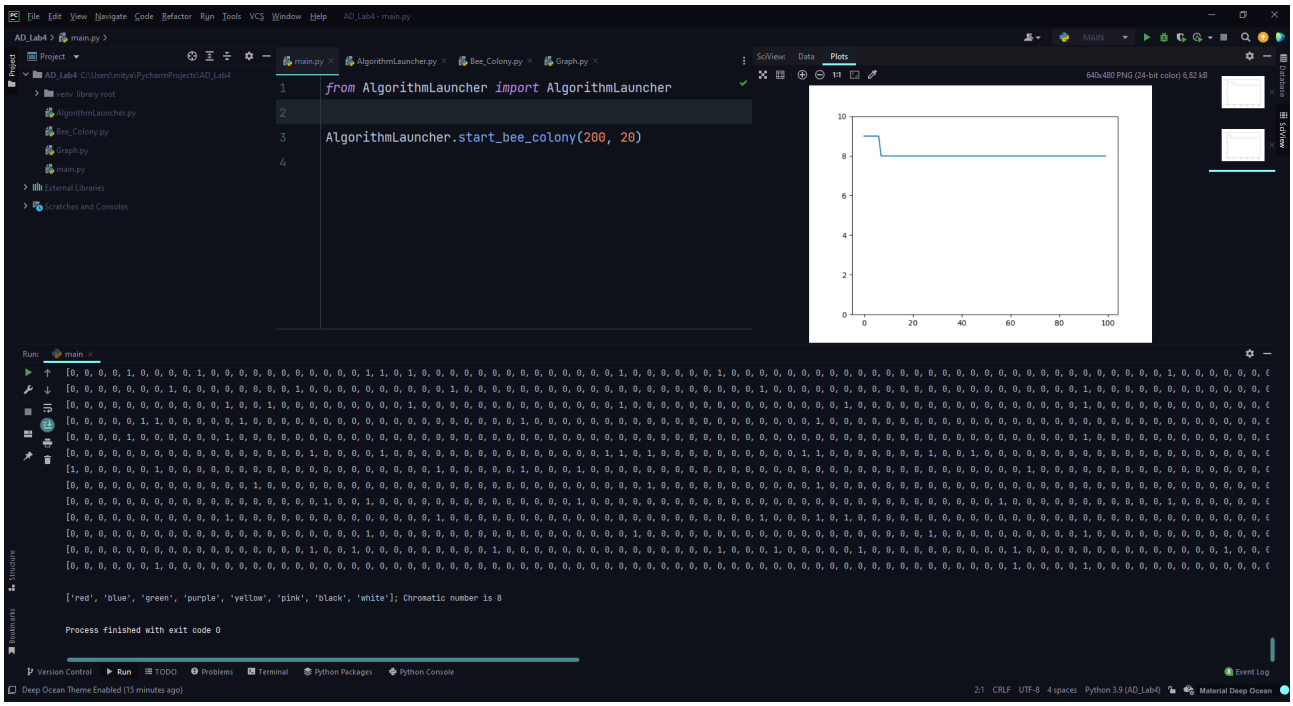

Рисунок 3.1 – Приклад роботи програми

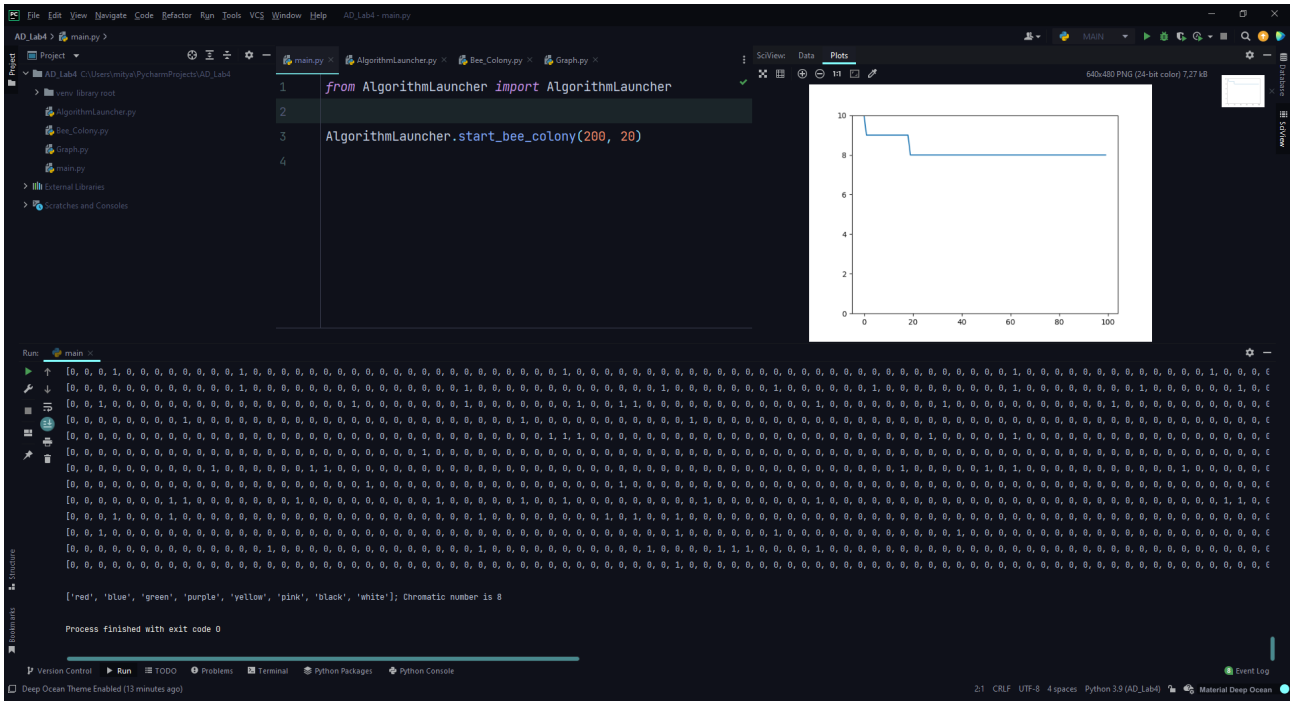

Рисунок 3.2 – Приклад роботи програми

3.2 Тестування алгоритму

3.2.1 Значення цільової функції зі збільшенням кількості ітерацій

У таблиці 3.1 наведено значення цільової функції зі збільшенням кількості ітерацій.

Кількість ітерацій	Значення цільової функції	Кількість ітерацій	Значення цільової функції
1	10	26	8
2	10	27	8
3	9	28	8
4	9	29	8
5	9	30	8
6	9	31	8
7	9	32	8
8	9	33	8
9	9	34	8
10	9	35	8
11	9	36	8
12	9	37	8
13	9	38	8
14	9	39	8
15	9	40	8
16	9	41	8
17	9	42	8
18	8	43	8
19	8	44	8
20	8	45	8
21	8	46	8
22	8	47	8
23	8	48	8
24	8	49	8
25	8	50	8

Таблиця 3.1 – Значення цільової функції зі збільшенням кількості ітерацій

3.2.2 Графіки залежності розв'язку від числа ітерацій

На рисунку 3.3 наведений графік, який показує якість отриманого розв'язку.

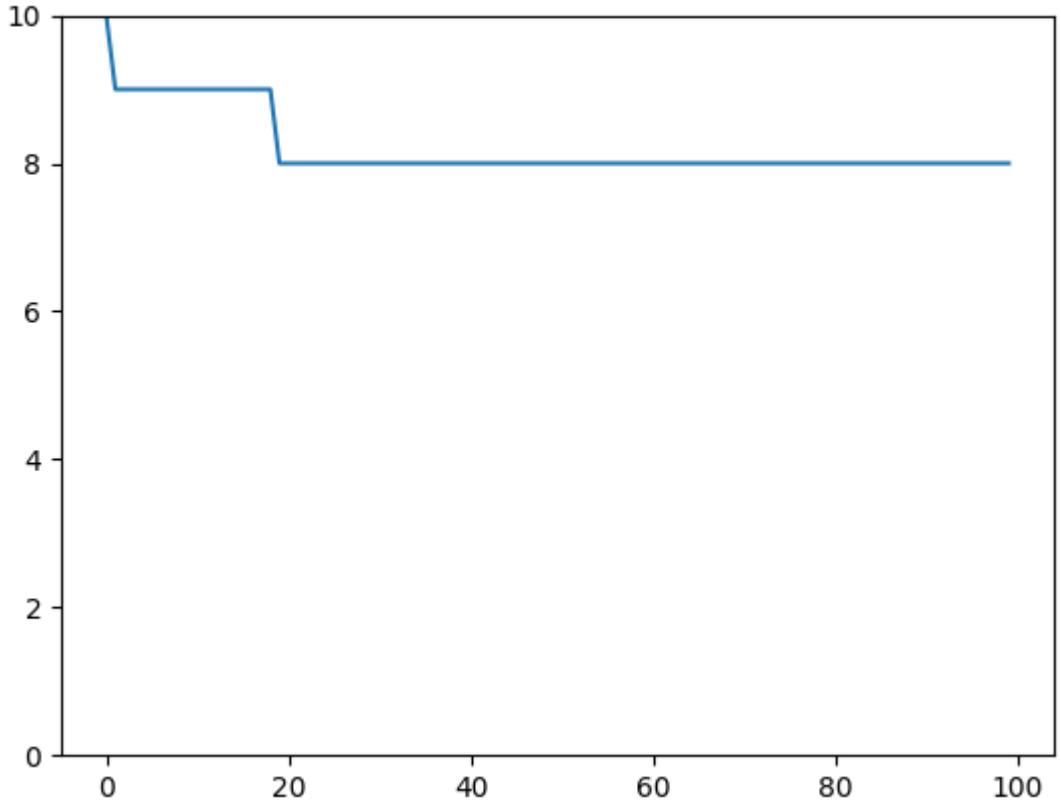

Рисунок 3.3 – Графік залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи вивчено основні підходи формалізації метаевристичних алгоритмів і вирішення типових задач з їхньою допомогою на прикладі бджолиного алгоритму ABC. Розроблено програмну реалізацію даного алгоритму, проаналізовано якість рішення поставлених задач, побудовано відповідні графіки.

Критерії оцінювання

При здачі лабораторної роботи до 27.11.2021 включно максимальний бал дорівнює – 5. Після 27.11.2021 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- програмна реалізація алгоритму – 75%;
- тестування алгоритму– 20%;
- висновок – 5%.