

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)

ІІІ-25 Карнов Л В
(шифр, прізвище, ім'я, по батькові)

Перевірив

Головченко М.Н.
(прізвище, ім'я, по батькові)

Київ 2023

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
2	ЗАВДАННЯ	4
3	ВИКОНАННЯ.....	10
3.1	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ	10
3.1.1	<i>Вихідний код.....</i>	<i>10</i>
3.1.2	<i>Приклади роботи</i>	<i>12</i>
3.2	ТЕСТУВАННЯ АЛГОРИТМУ	15
3.2.1	<i>Значення цільової функції зі збільшенням кількості ітерацій .</i>	<i>15</i>
3.2.2	<i>Графіки залежності розв'язку від числа ітерацій</i>	<i>15</i>
	ВИСНОВОК	17
	КРИТЕРІЇ ОЦІНЮВАННЯ	18

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

№	Задача і алгоритм
1	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
2	Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 30$, починають маршрут в різних випадкових вершинах).
3	Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники).
4	Задача про рюкзак (місткість $P=200$, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити

	власний оператор локального покращення.
5	Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, L_{min} знайти жадібним алгоритмом, кількість мурах $M = 35$, починають маршрут в різних випадкових вершинах).
6	Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники).
7	Задача про рюкзак (місткість $P=150$, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
8	Задача комівояжера (200 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм ($\alpha = 3$, $\beta = 2$, $\rho = 0,3$, L_{min} знайти жадібним алгоритмом, кількість мурах $M = 45$, починають маршрут в різних випадкових вершинах).
9	Задача розфарбовування графу (150 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 25 із них 3 розвідники).
10	Задача про рюкзак (місткість $P=150$, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
11	Задача комівояжера (250 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho =$

	0,6, L_{min} знайти жадібним алгоритмом, кількість мурах $M = 45$, починають маршрут в різних випадкових вершинах).
12	Задача розфарбовування графу (300 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 60 із них 5 розвідники).
13	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий 30% і 70%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
14	Задача комівояжера (250 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм ($\alpha = 4$, $\beta = 2$, $\rho = 0,3$, L_{min} знайти жадібним алгоритмом, кількість мурах $M = 45$ (10 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових вершинах).
15	Задача розфарбовування графу (100 вершин, степінь вершини не більше 20, але не менше 1), класичний бджолиний алгоритм (число бджіл 30 із них 3 розвідники).
16	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий 30%, 40% і 30%, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
17	Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0,7$, L_{min} знайти жадібним алгоритмом, кількість мурах $M = 45$ (15 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових

	вершинах).
18	Задача розфарбовування графу (300 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 60 із них 5 розвідники).
19	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
20	Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм ($\alpha = 3$, $\beta = 2$, $\rho = 0,7$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 45$ (10 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах).
21	Задача розфарбовування графу (200 вершин, степінь вершини не більше 30, але не менше 1), класичний бджолиний алгоритм (число бджіл 40 із них 2 розвідники).
22	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
23	Задача комівояжера (300 вершин, відстань між вершинами випадкова від 1 до 60), мурашиний алгоритм ($\alpha = 3$, $\beta = 2$, $\rho = 0,6$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 45$ (15 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах).

24	Задача розфарбовування графу (400 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 70 із них 10 розвідники).
25	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
26	Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 30$, починають маршрут в різних випадкових вершинах).
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).
28	Задача про рюкзак (місткість $P=200$, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
29	Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 35$, починають маршрут в різних випадкових вершинах).
30	Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники).

31	Задача про рюкзак (місткість $P=250$, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
32	Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 30$, починають маршрут в різних випадкових вершинах).
33	Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники).
34	Задача про рюкзак (місткість $P=200$, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
35	Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 35$, починають маршрут в різних випадкових вершинах).

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
import copy
import random
from copy import deepcopy

import networkx as nx
import matplotlib.pyplot as plt
from matplotlib import pylab

GRAPH_SIZE = 150
EDGE_PROBABILITY = 0.05

BEE_COUNT = 22
SCOUT_COUNT = 3

COLORS = [None, 'green', 'red', 'yellow', 'blue', 'purple', 'darkred', 'orange',
          'lime', 'cyan', 'violet']

class Cell:
    def __init__(self, val: bool = 0):
        self.val = val

    def __repr__(self):
        return str(self.val)

    def __bool__(self):
        return bool(self.val)

class Graph:
    def __init__(self, vertex_count: int = GRAPH_SIZE, skip_rand_gen: bool =
False):
        self.count = vertex_count

        if skip_rand_gen:
            return
        self._edge_table = [
            [Cell(random.randint(0, 99) < (EDGE_PROBABILITY * 100)) for _ in
range(i)]
            for i in range(vertex_count)
        ]

        self.color_map = [0 for _ in range(vertex_count)]
        self.used_colors = set()
        self.visited = set()

    def print_table(self):
        print(*self._edge_table, sep='\n')

    def edges(self, vertex):
        res = []
        if vertex != 0:
            res += enumerate(self._edge_table[vertex])
```

```

        for i in range(vertex, self.count - 1):
            res.append((i + 1, self._edge_table[i + 1][vertex]))

        return res

    def adjacent_vertexes(self, vertex):
        return [vert for vert, val in self.edges(vertex) if val]

    def adjacent_vertexes_colors(self, vertex):
        return set([self.color_map[vert] for vert in
self.adjacent_vertexes(vertex)])

    def power_of_vertex(self, vertex):
        return sum(map(lambda x: x[1].val, self.edges(vertex)))

    def draw(self):
        visual = []

        for i in range(self.count):
            for j in range(i):
                if self._edge_table[i][j].val:
                    visual.append([i, j])

        G = nx.Graph()
        G.add_nodes_from([i for i in range(self.count)])

        G.add_edges_from(visual)
        plt.figure(figsize=(32, 24))
        # plt.axis = False
        color_map = [COLORS[i] for i in self.color_map]
        nx.draw_networkx(G, node_color=color_map, with_labels=True)
        plt.show()

    @property
    def chromatic_number(self):
        used_colors = set()
        for i in range(self.count):
            used_colors.add(self.color_map[i])

        return len(used_colors)

    def color_graph(self):
        while not all(self.color_map):
            pop = [i for i in range(self.count) if i not in self.visited]
            taken_vertexes = random.sample(pop, SCOUT_COUNT) if len(pop) >
SCOUT_COUNT else pop
            nectar = [self.power_of_vertex(i) for i in taken_vertexes]
            if sum(nectar) == 0:
                self.color_vertex(random.choice(taken_vertexes))
                continue
            to_visit = random.choices(taken_vertexes, nectar)[0]
            self.visited.add(to_visit)

            self.color_vertex(to_visit)

    def color_vertex(self, vertex):
        self.color_map[vertex] = 0
        if self.power_of_vertex(vertex) == 0:
            self.color_map[vertex] = 1
            self.used_colors.add(1)
        else:
            adjacent = self.adjacent_vertexes(vertex)
            if BEE_COUNT < len(adjacent):
                nxt = random.sample(adjacent, BEE_COUNT)

```

```

        else:
            nxt = adjacent

        for nvert in nxt:
            for color in range(1, self.count):
                if color not in self.adjacent_vertexes_colors(nvert):
                    self.used_colors.add(color)
                    self.color_map[nvert] = color
                    break
            for color in range(1, self.count):
                if color not in self.adjacent_vertexes_colors(vertex):
                    self.used_colors.add(color)
                    self.color_map[vertex] = color
                    break

    def improve_coloring(self, iteration_count: int):
        self.visited = set()
        for _ in range(iteration_count):
            vertexes = [i for i in range(self.count) if i not in self.visited]
            nectar = [self.power_of_vertex(i) for i in vertexes]
            if len(vertexes) == 0:
                return
            if sum(nectar) == 0:
                visited = random.choice(vertexes)
            else:
                visited = random.choices(vertexes, nectar)[0]
            self.color_vertex(visited)
            self.visited.add(visited)

    def copy(self):
        new = Graph(self.count)
        new._edge_table = copy.deepcopy(self._edge_table)
        new.color_map = copy.deepcopy(self.color_map)

        return new

def _main():
    g = Graph()
    g.color_graph()
    print('Chromatic number: ', g.chromatic_number)
    g.draw()
    states = []
    for _ in range(10):
        states.append(g)
        g = g.copy()
        g.improve_coloring(10)
        print('Chromatic number: ', g.chromatic_number)
        g.draw()

if __name__ == '__main__':
    _main()

```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

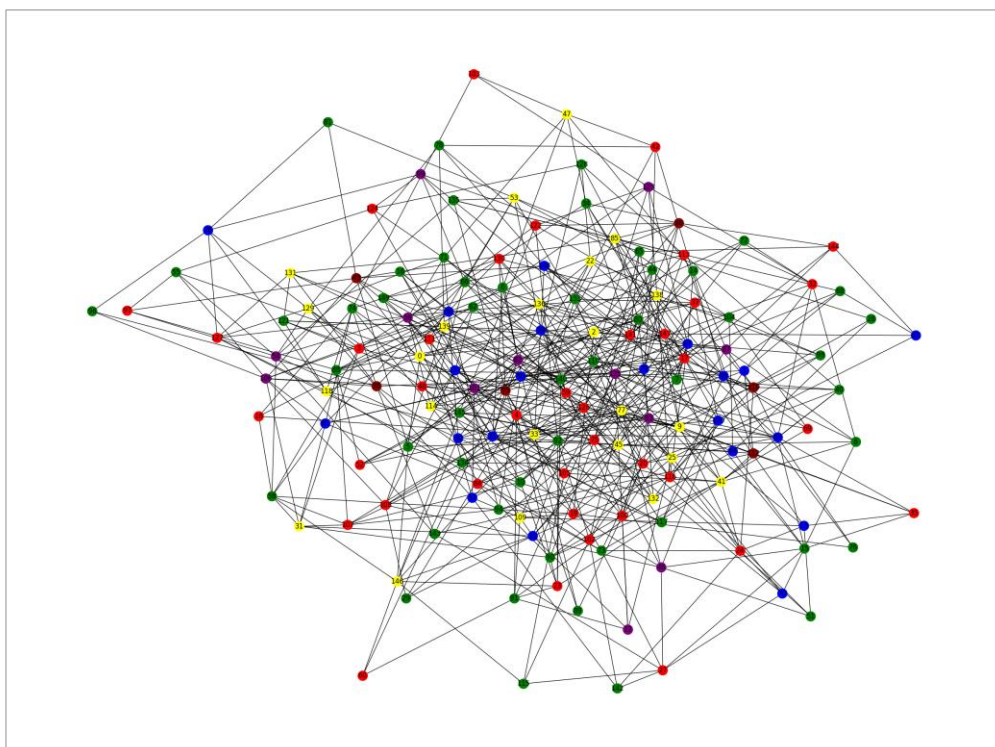


Рисунок 3.1 –

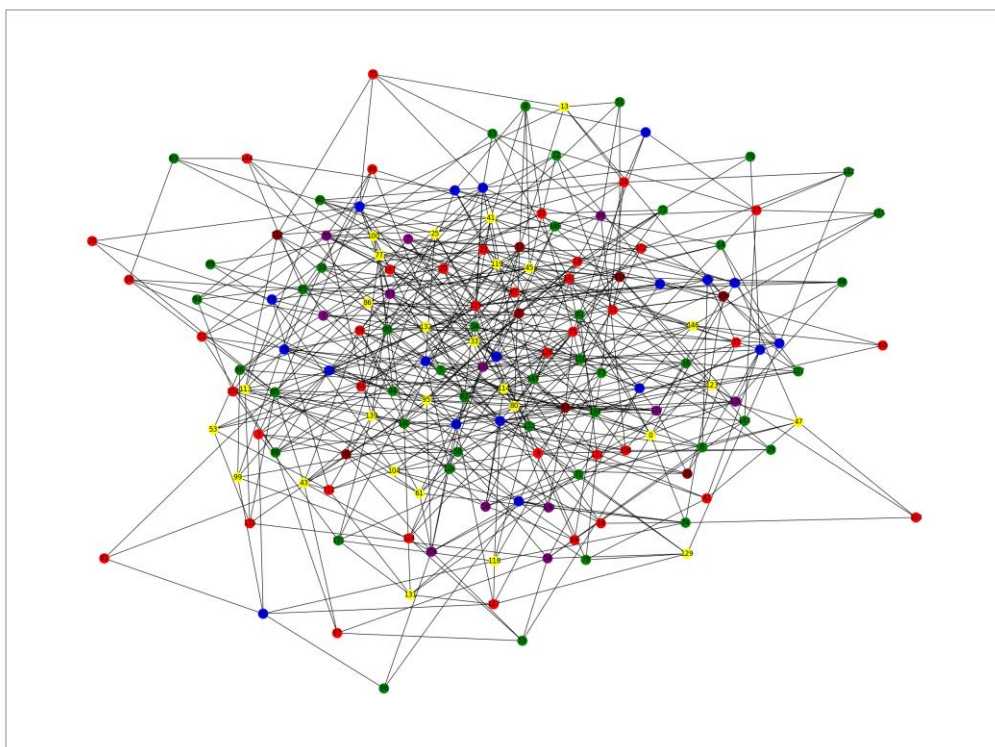


Рисунок 3.2 –

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.

iterations	Chromatic number
0	7
10	7
20	6
30	7
40	7
50	7
60	6

3.2.2 Графіки залежності розв'язку від числа ітерацій

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

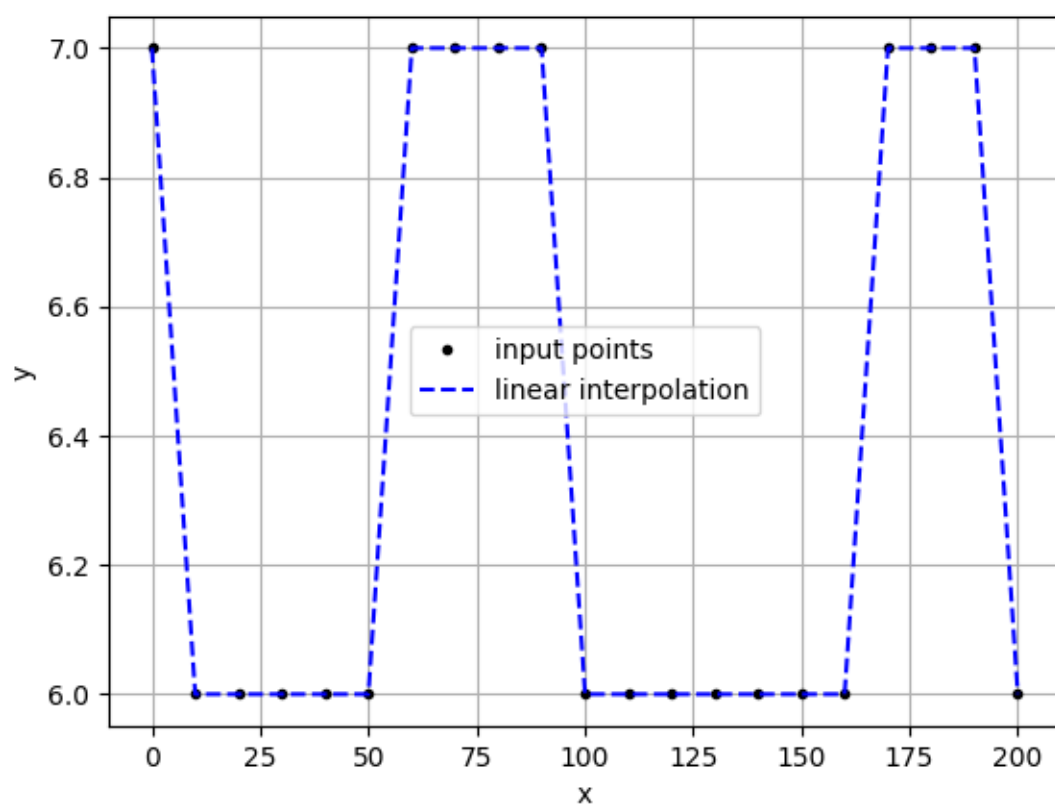


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

ВИСНОВОК

В рамках даної лабораторної роботи я розробив алгоритм розфарбування графа використовуючи симуляцію штучної колонії бджіл. Цей підхід забезпечує непогону точність в початкової стадії, а ітеративна стадія може як покращити результат так і погіршити його, тому якщо після останніх ітерацій хроматичне число зростало програма відновлює минулий стан та пробує знову. З спостережень амплітуда змінення хроматичного числа не перебільшувала 1, тому можна з впевненістю після декількох ітерацій визначити хроматичне число.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 10.12.2023 включно максимальний бал дорівнює – 5. Після 10.12.2023 максимальний бал дорівнює – 4,5.

Критерії оцінювання у відсотках від максимального балу:

- програмна реалізація алгоритму – 55%;
- робота з гіт – 20%;
- тестування алгоритму – 20%;
- висновок – 5%.

+1 додатковий бал можна отримати за виконання роботи до 3.12.2023