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”
Варіант 24

Виконав(ла) *ІП-12 Титаренко Данило* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов О. О.* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, Lmin 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 (α = 2, β = 3, ρ = 0,4, Lmin 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 (α = 3, β = 2, ρ = 0,3, Lmin 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 (α = 2, β = 4, ρ

	= 0,6, Lmin 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 (α = 4, β = 2, ρ = 0,3, Lmin 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 (α = 2, β = 4, ρ = 0,7, Lmin 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 (α = 3, β = 2, ρ = 0,7, Lmin 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 (α = 3, β = 2, ρ = 0,6, Lmin 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, Lmin 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 ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, Lmin 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, Lmin 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 ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 35, починають маршрут в різних випадкових вершинах).

3 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
from random import shuffle
from queue import PriorityQueue
import networkx as nx
import matplotlib.pyplot as plt


SCOUTS_BEE = 10
BEST_SCOUTS_BEE = 5
RANDOM_SCOUTS_BEE = SCOUTS_BEE - BEST_SCOUTS_BEE
FORAGERS_BEE = 60
BEST_FORAGERS_BEE = 30
RANDOM_FORAGERS_BEE = 5


class Bee:
    def __init__(self,color_node,f):
        self.f = f
        self.color_node = color_node


    def __lt__(self, other):
        return self.f.__lt__(other.f)

    def __repr__(self):
        return f"\nChromatic number: {self.f} Coloring: {self.color_node}"

# створення випадкового графу
def create_graph(nodes,max_power,min_power):
    g = nx.watts_strogatz_graph(nodes, max_power, min_power)
    graph = {}
    for item in g.edges:
        node, neighbour = item
        node += 1
        neighbour += 1
        if node not in graph:
            graph[node] = []
        if neighbour not in graph:
            graph[neighbour] = []
        graph[node].append(neighbour)
        graph[neighbour].append(node)
    return graph,g
# намалювати граф
def show_graph(color_node,graph):
```

```
    colors = ['red', 'blue', 'yellow', 'lime', 'orange', 'black', 'green', 'grey',
'purple', 'pink','brown','maroon','aqua','azure','teal','wheat','navy','salmon']
    new_colors=[]
    for item in list(color_node.values()):
        new_colors.append(colors[item-1])
    nx.draw(graph, with_labels=True,node_color=list(new_colors))
    plt.show()
# початкове розфарбування графа
def greedy(graph):
    color_node = {node: 0 for node in graph.keys()}
    graph = list(graph.items())
    shuffle(graph)
    number_color = 0
    for node, neighbours in graph:
        for color in range(1, 100):
            if not check_neighbour_color(color, neighbours, color_node):
                color_node[node] = color
                if color > number_color:
                    number_color += 1
                break
    return color_node, number_color
# Перевірити колір вершини на співпадіння кольорів вершин її сусідів
def check_neighbour_color(color, neighbours, color_node):
    if neighbours:
        for neighbour in neighbours:
            if neighbour in color_node and color_node[neighbour] == color:
                return True
    return False


def create_scouts(graph, bee_scouts, n):
    while bee_scouts.qsize() < n:
        scout = Bee(*greedy(graph))
        if scout not in bee_scouts.queue:
            bee_scouts.put(scout)



# Основна частина бджолинного алгоритму
def bee_optimization(bee_scout, graph, bee_local_foragers):
    queue = []

    for node, neighbours in sorted(list(graph.items()), key=lambda x: len(x[1]),
reverse=True):
        if len(queue) >= bee_local_foragers:
            break
        for neighbour in neighbours:
            queue.append((node, neighbour))

    results = []
```

```
    for node, neighbour in queue:
        new_color_node = dict(bee_scout.color_node)
        new_color_node[neighbour], new_color_node[node] = new_color_node[node],
new_color_node[neighbour]

        if check_neighbour_color(new_color_node[node], graph[node], new_color_node)
or \
                check_neighbour_color(new_color_node[neighbour], graph[neighbour],
new_color_node):
            continue
        else:
            for color in range(1, bee_scout.f + 1):
                if not check_neighbour_color(color, graph[neighbour],
new_color_node):
                    new_color_node[neighbour] = color
                    results.append(Bee(new_color_node,
len(set(new_color_node.values()))))

    return min(results, key=lambda x: x.f) if results else bee_scout

def local_search(graph, bee_new_scouts, bee_scouts, n):
    for scout in bee_scouts:
        scout = bee_optimization(scout, graph, n)
        bee_new_scouts.put(scout)

def get_best_scouts(bee_scouts):
    best = []
    for i in range(BEST_SCOUTS_BEE):
        best.append(bee_scouts.get())
    return best

def add_best_scouts(bee_scouts, bee_best_scouts):
    for scout in bee_best_scouts:
        bee_scouts.put(scout)




def main():
    graph,g = create_graph(400,50,1)

    # Створення початкових бджіл-розвідників
    bee_scouts = PriorityQueue()
    create_scouts(graph, bee_scouts, SCOUTS_BEE)

    for i in range(1000):
        if  (i % 20==0):
            best = bee_scouts.get()
            print(f"i: {i}, min color_number: {best.f}")
            bee_scouts.put(best)
```

```
        bee_best_scouts = get_best_scouts(bee_scouts)
        bee_random_scouts = bee_scouts.queue

        bee_scouts = PriorityQueue()
        local_search(graph, bee_scouts, bee_best_scouts, BEST_FORAGERS_BEE)
        local_search(graph, bee_scouts, bee_random_scouts, RANDOM_FORAGERS_BEE)

    #    Зберігаємо найкращий результат та шукаємо нові рішення
        bee_best_scouts = get_best_scouts(bee_scouts)
        bee_scouts = PriorityQueue()
        create_scouts(graph, bee_scouts, RANDOM_SCOUTS_BEE)
        add_best_scouts(bee_scouts, bee_best_scouts)

    best = bee_scouts.get()
    color_node = {k: v for k, v in sorted(best.coloring.items(), key=lambda item:
item[0])}
    print(f"Best coloring: {best.f}\nNumber of colors: {color_node}")
    print(show_graph(color_node,g))

if __name__ == "__main__":
    main()
```

3.1.2 Приклади роботи

```
(.venv) PS C:\Users\Danylo\Desktop\python> & c:/Users/Danylo/Desktop/python/.venv/Scripts/python.exe c:/Users/Danylo/Desktop/python/main2.py
i: 0, min color_number: 18
i: 20, min color_number: 17
i: 40, min color_number: 17
i: 60, min color_number: 17
i: 80, min color_number: 17
i: 100, min color_number: 17
i: 120, min color_number: 16
i: 140, min color_number: 16
i: 160, min color_number: 16
i: 180, min color_number: 16
i: 200, min color_number: 16
i: 220, min color_number: 16
i: 240, min color_number: 16
i: 260, min color_number: 16
i: 280, min color_number: 16
i: 300, min color_number: 16
i: 320, min color_number: 16
i: 340, min color_number: 16
i: 360, min color_number: 16
i: 380, min color_number: 16
i: 400, min color_number: 16
i: 420, min color_number: 16
i: 440, min color_number: 16
i: 460, min color_number: 16
i: 480, min color_number: 16
i: 500, min color_number: 16
i: 520, min color_number: 16
i: 540, min color_number: 16
i: 560, min color_number: 16
i: 580, min color_number: 16
i: 600, min color_number: 16
i: 620, min color_number: 16
i: 640, min color_number: 16
i: 660, min color_number: 16
i: 680, min color_number: 16
i: 700, min color_number: 16
i: 720, min color_number: 16
i: 740, min color_number: 16
i: 760, min color_number: 16
i: 780, min color_number: 16
i: 800, min color_number: 16
i: 820, min color_number: 16
i: 840, min color_number: 16
```

Рисунок 3.1 – Приклад роботи програми для випадкового графу

```
i: 860, min color_number: 16
i: 880, min color_number: 16
i: 900, min color_number: 16
i: 920, min color_number: 16
i: 940, min color_number: 16
i: 960, min color_number: 16
i: 980, min color_number: 16
Best coloring: 16
Number of colors: {1: 15, 2: 3, 3: 9, 4: 15, 5: 2, 6: 13, 7: 13, 8: 8, 9: 8, 10: 16, 11: 3, 12: 7, 13: 14, 14: 5, 15: 12, 16: 11, 17: 10, 18: 1, 19: 2, 20: 9, 21: 15,
 22: 13, 23: 10, 24: 10, 25: 6, 26: 11, 27: 2, 28: 10, 29: 14, 30: 5, 31: 5, 32: 12, 33: 4, 34: 6, 35: 5, 36: 3, 37: 1, 38: 7, 39: 7, 40: 8, 41: 11, 42: 9, 43: 1, 44:
 12, 45: 13, 46: 10, 47: 14, 48: 16, 49: 5, 50: 8, 51: 8, 52: 7, 53: 3, 54: 1, 55: 3, 56: 14, 57: 4, 58: 10, 59: 13, 60: 9, 61: 4, 62: 3, 63: 2, 64: 10, 65: 6, 66: 3,
 67: 4, 68: 1, 69: 16, 70: 7, 71: 3, 72: 7, 73: 8, 74: 15, 75: 11, 76: 3, 77: 4, 78: 3, 79: 9, 80: 6, 81: 2, 82: 2, 83: 1, 84: 8, 85: 13, 86: 11, 87: 5, 88: 10, 89: 4
, 90: 14, 91: 16, 92: 5, 93: 4, 94: 8, 95: 12, 96: 11, 97: 5, 98: 11, 99: 6, 100: 13, 101: 11, 102: 12, 103: 4, 104: 7, 105: 4, 106: 1, 107: 5, 108: 9, 109: 9, 110: 7
, 111: 1, 112: 7, 113: 13, 114: 11, 115: 13, 116: 8, 117: 1, 118: 12, 119: 9, 120: 11, 121: 15, 122: 8, 123: 10, 124: 7, 125: 1, 126: 4, 127: 7, 128: 5, 129: 4, 130:
8, 131: 13, 132: 9, 133: 5, 134: 3, 135: 12, 136: 15, 137: 2, 138: 14, 139: 7, 140: 3, 141: 4, 142: 12, 143: 6, 144: 5, 145: 13, 146: 15, 147: 2, 148: 10, 149: 11, 15
0: 3, 151: 6, 152: 7, 153: 7, 154: 16, 155: 14, 156: 11, 157: 4, 158: 1, 159: 8, 160: 7, 161: 6, 162: 12, 163: 9, 164: 1, 165: 4, 166: 12, 167: 12, 168: 9, 169: 6, 17
0: 13, 171: 13, 172: 10, 173: 7, 174: 10, 175: 10, 176: 2, 177: 11, 178: 6, 179: 10, 180: 6, 181: 9, 182: 4, 183: 3, 184: 8, 185: 4, 186: 14, 187: 6, 188: 5, 189: 4,
190: 10, 191: 12, 192: 4, 193: 3, 194: 1, 195: 4, 196: 2, 197: 8, 198: 3, 199: 3, 200: 14, 201: 5, 202: 7, 203: 4, 204: 2, 205: 14, 206: 8, 207: 4, 208: 4, 209: 5, 21
0: 3, 211: 15, 212: 2, 213: 1, 214: 3, 215: 10, 216: 2, 217: 3, 218: 8, 219: 6, 220: 4, 221: 7, 222: 10, 223: 11, 224: 6, 225: 9, 226: 5, 227: 9, 228: 2, 229: 4, 230:
 14, 231: 2, 232: 11, 233: 7, 234: 7, 235: 1, 236: 2, 237: 4, 238: 6, 239: 7, 240: 2, 241: 3, 242: 8, 243: 4, 244: 2, 245: 1, 246: 10, 247: 11, 248: 5, 249: 5, 250: 6
, 251: 6, 252: 3, 253: 9, 254: 5, 255: 13, 256: 16, 257: 5, 258: 8, 259: 12, 260: 3, 261: 9, 262: 5, 263: 11, 264: 12, 265: 13, 266: 1, 267: 2, 268: 2, 269: 12, 270:
4, 271: 6, 272: 6, 273: 6, 274: 12, 275: 6, 276: 6, 277: 13, 278: 3, 279: 5, 280: 5, 281: 8, 282: 10, 283: 13, 284: 12, 285: 1, 286: 2, 287: 2, 288: 8, 289: 9, 290: 3
, 291: 7, 292: 16, 293: 11, 294: 11, 295: 6, 296: 8, 297: 2, 298: 13, 299: 11, 300: 1, 301: 1, 302: 6, 303: 3, 304: 5, 305: 9, 306: 3, 307: 16, 308: 11, 309: 1, 310:
14, 311: 13, 312: 4, 313: 6, 314: 13, 315: 14, 316: 2, 317: 1, 318: 14, 319: 1, 320: 6, 321: 14, 322: 3, 323: 13, 324: 8, 325: 6, 326: 4, 327: 2, 328: 6, 329: 12, 330
: 1, 331: 8, 332: 7, 333: 9, 334: 7, 335: 12, 336: 16, 337: 2, 338: 12, 339: 9, 340: 2, 341: 12, 342: 1, 343: 9, 344: 5, 345: 1, 346: 11, 347: 4, 348: 15, 349: 7, 350
: 15, 351: 3, 352: 2, 353: 11, 354: 2, 355: 9, 356: 9, 357: 7, 358: 1, 359: 2, 360: 5, 361: 3, 362: 5, 363: 8, 364: 2, 365: 11, 366: 11, 367: 1, 368: 9, 369: 15, 370:
 10, 371: 6, 372: 8, 373: 16, 374: 6, 375: 6, 376: 2, 377: 7, 378: 8, 379: 1, 380: 2, 381: 15, 382: 9, 383: 10, 384: 9, 385: 5, 386: 1, 387: 13, 388: 2, 389: 3, 390:
8, 391: 4, 392: 14, 393: 7, 394: 8, 395: 10, 396: 5, 397: 4, 398: 7, 399: 7, 400: 5}
```

Рисунок 3.2 – Приклад роботи програми для випадкового графу

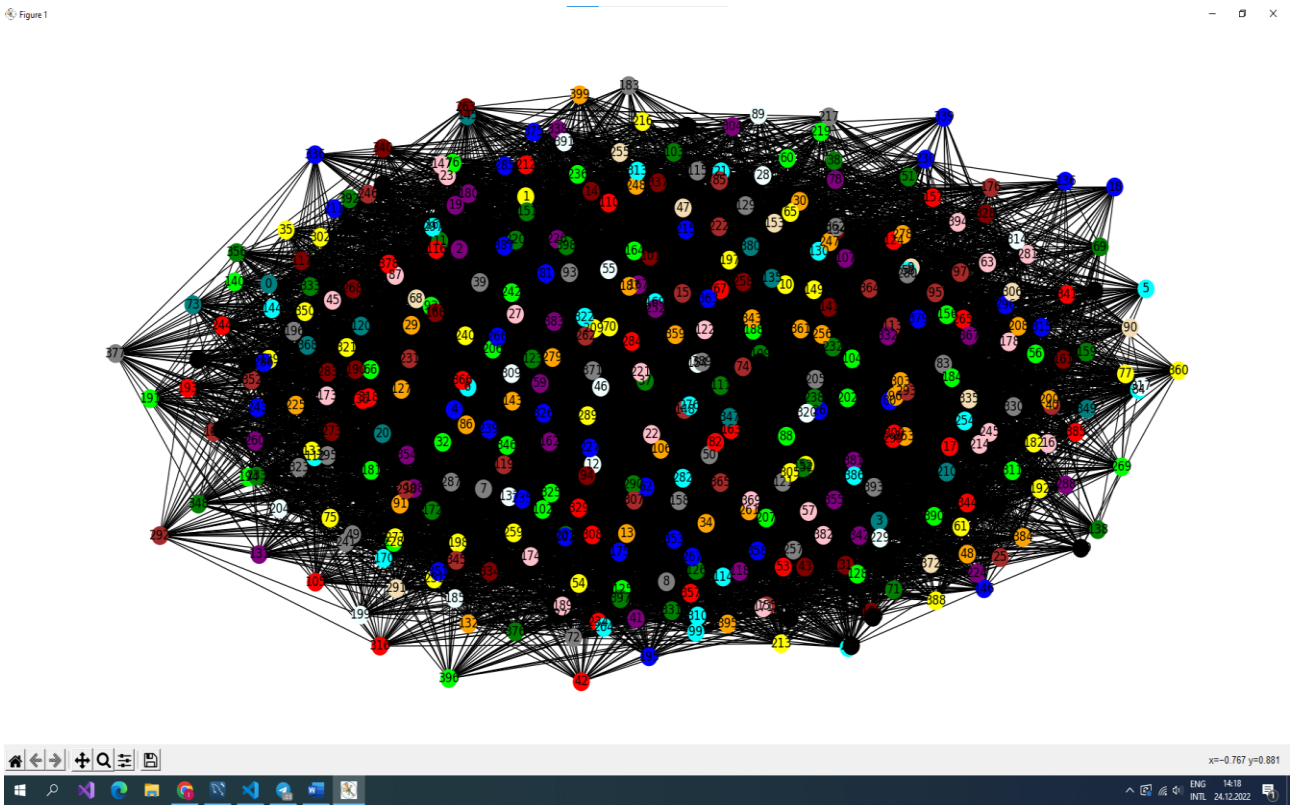

Рисунок 3.3 – Приклад роботи програми для випадкового графу

3.2 Тестування алгоритму

3.2.1 Значення цільової функції зі збільшенням кількості ітерацій

Ітерація	Значення функції	Ітерація	Значення функції
0	18	500	16
20	17	520	16
40	17	540	16
60	17	560	16
80	17	580	16
100	17	600	16
120	16	620	16
140	16	640	16
160	16	660	16
180	16	680	16
200	16	700	16
220	16	720	16
240	16	740	16
260	16	760	16
280	16	780	16
300	16	800	16
320	16	820	16
340	16	840	16
360	16	860	16
380	16	880	16
400	16	900	16
420	16	920	16
440	16	940	16
460	16	960	16
480	16	980	16

3.2.2 Графіки залежності розв'язку від числа ітерацій

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

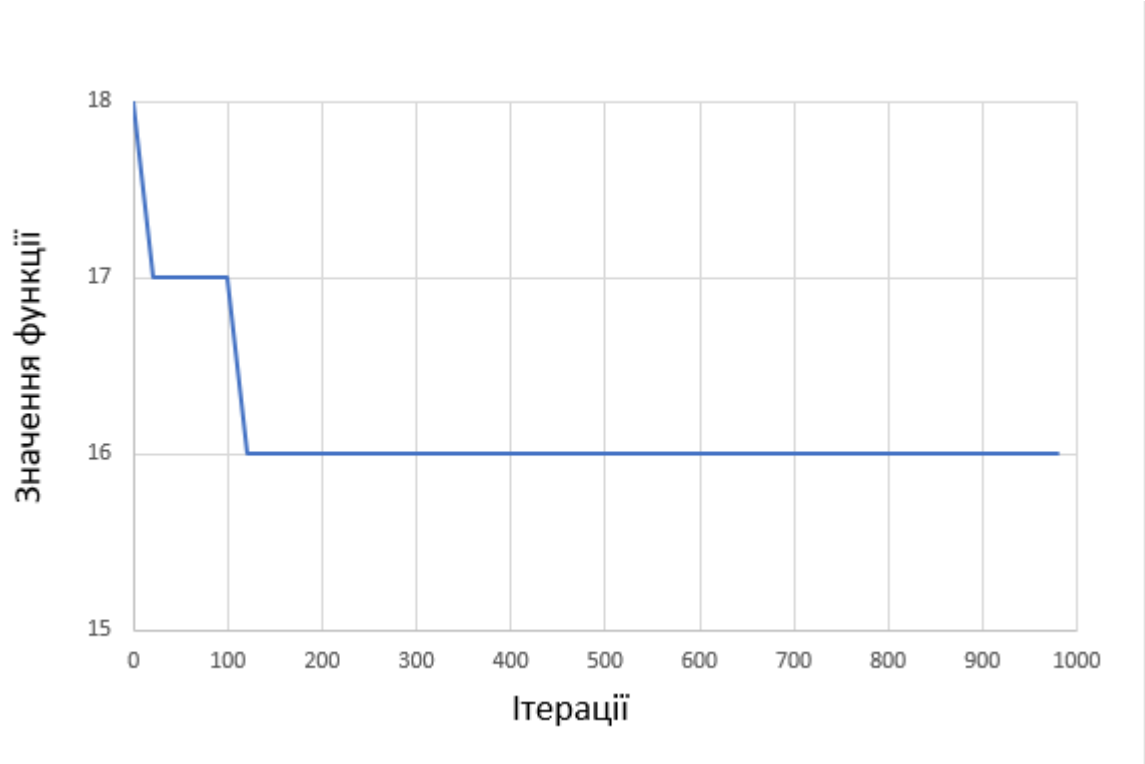

Рисунок 3.4 – Графіки залежності розв'язку від числа ітерацій

ВИСНОВОК

В рамках даної лабораторної роботи було досліджено основні підходи формалізації метаевристичних алгоритмів, а саме класичний бджолиний алгоритм. Також за допомогою даного алгоритму було розв’язано задачу про розфарбування графу, який має 400 вершин та максимальну степінь 50. Особливістю мого алгоритму було використання 70 бджіл, 10 з яких – розвідники. Задачу було реалізовано на мові програмування Python, в ході виконанні програми було зменшено хроматичне число(кількість кольорів) графа від 18 до 16, що є непоганим результатом

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%.