**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)**

(шифр, прізвище, ім'я, по батькові)

*ІП-25 Карпов Л В*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2023

Зміст

[1 Мета лабораторної роботи 3](#_Toc51260917)

[2 Завдання 4](#_Toc51260918)

[3 Виконання 10](#_Toc51260919)

[3.1 Програмна реалізація алгоритму 10](#_Toc51260920)

[3.1.1 Вихідний код 10](#_Toc51260921)

[3.1.2 Приклади роботи 12](#_Toc51260922)

[3.2 Тестування алгоритму 15](#_Toc51260923)

[3.2.1 Значення цільової функції зі збільшенням кількості ітерацій 15](#_Toc51260924)

[3.2.2 Графіки залежності розв'язку від числа ітерацій 15](#_Toc51260925)

[Висновок 17](#_Toc51260926)

[Критерії оцінювання 18](#_Toc51260927)

# Мета лабораторної роботи

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

# 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 (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 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 М = 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 М = 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 М = 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 М = 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 М = 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 М = 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 М = 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 (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 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 (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 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 (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 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 (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 35, починають маршрут в різних випадкових вершинах). |

# Виконання

## Програмна реалізація алгоритму

### 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()

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

A group of colorful dots and lines

Description automatically generatedРисунок 3.1 –

A group of colorful dots and lines

Description automatically generated

Рисунок 3.2 –

## Тестування алгоритму

### 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 |

### Графіки залежності розв'язку від числа ітерацій

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

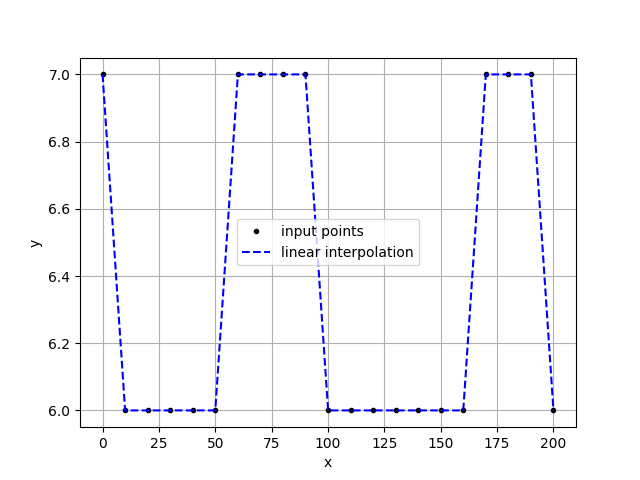


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

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 покращети результат так і погіршити його, тому якщо після останіх 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 вмзначити 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