**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-21 Скрипець Ольга*

**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 10](#_Toc51260922)

[3.2 Тестування алгоритму 11](#_Toc51260923)

[3.2.1 Значення цільової функції зі збільшенням кількості ітерацій 11](#_Toc51260924)

[3.2.2 Графіки залежності розв'язку від числа ітерацій 11](#_Toc51260925)

[Висновок 12](#_Toc51260926)

[Критерії оцінювання 13](#_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 json  
import random  
  
class TravelingSalesmanProblem:  
 FILE\_NAME = "problem.json"  
  
 def \_\_init\_\_(self):  
 self.initialize\_matrix()  
 self.optimal\_solution = self.find\_optimal\_solution()  
  
 def get\_cost(self, path):  
 solution = 0  
 for i in range(100):  
 solution += self.get\_distance(path[i], path[i + 1])  
 return solution  
  
 def get\_distance(self, source, destination):  
 return self.matrix[source][destination]  
  
 def initialize\_matrix(self):  
 try:  
 with open(self.FILE\_NAME, 'r') as file:  
 self.matrix = json.load(file)  
 except FileNotFoundError:  
 self.matrix = [[0 if i == j else random.randint(5, 50) for j in range(100)] for i in range(100)]  
 with open(self.FILE\_NAME, 'w') as file:  
 json.dump(self.matrix, file)  
  
 def find\_optimal\_solution(self):  
 solutions = []  
 for j in range(100):  
 nodes = list(range(100))  
 current\_node = j  
 path = [current\_node]  
 for i in range(99):  
 node = current\_node  
 current\_node = min(filter(lambda x: x not in path, nodes), key=lambda x: self.get\_distance(node, x))  
 path.append(current\_node)  
 path.append(j)  
 solutions.append(self.get\_cost(path))  
  
 return min(solutions)  
  
  
class AntAlgorithm:  
 def \_\_init\_\_(self, problem):  
 self.problem = problem  
 self.pheromone\_matrix = self.initialize\_matrix()  
  
 def iterate(self):  
 paths = [self.find\_path(random.randint(0, 99)) for \_ in range(30)]  
 self.update\_pheromones(paths)  
  
 def get\_solution(self):  
 path = [0]  
 for i in range(1, 100):  
 current\_index = path[i - 1]  
 pheromone\_array = self.pheromone\_matrix[current\_index]  
 max\_pheromone = float('-inf')  
 max\_index = -1  
 for j in range(1, 100):  
 if j not in path:  
 if pheromone\_array[j] > max\_pheromone:  
 max\_pheromone = pheromone\_array[j]  
 max\_index = j  
 path.append(max\_index)  
  
 path.append(0)  
 return path  
  
 def get\_probabilities(self, current\_node, allowed\_nodes):  
 values = [(self.pheromone\_matrix[current\_node][allowed\_node] \*\* 2) \* (  
 1.0 / self.problem.matrix[current\_node][allowed\_node] \*\* 4)  
 for allowed\_node in allowed\_nodes]  
 total = sum(values)  
  
 if total == 0:  
 return [0] \* len(values)  
  
 probabilities = [value / total for value in values]  
 return probabilities  
  
 def update\_pheromones(self, paths):  
 evaporation\_rate = 1-0.4  
 for i in range(100):  
 for j in range(100):  
 self.pheromone\_matrix[i][j] \*= evaporation\_rate  
  
 for path in paths:  
 cost = self.problem.get\_cost(path)  
 for i in range(100):  
 self.pheromone\_matrix[path[i]][path[i + 1]] += self.problem.optimal\_solution / cost  
  
 def find\_path(self, initial):  
 result = [initial]  
 to\_visit = list(range(100))  
 to\_visit.remove(initial)  
 for i in range(1, 100):  
 probabilities = self.get\_probabilities(result[i - 1], to\_visit)  
 node\_index = self.choose\_node(probabilities)  
 result.append(to\_visit[node\_index])  
 to\_visit.pop(node\_index)  
  
 result.append(initial)  
 return result  
  
 @staticmethod  
 def choose\_node(probabilities):  
 random\_value = random.random()  
 total = 0.0  
 for i, prob in enumerate(probabilities):  
 total += prob  
 if total > random\_value:  
 return i  
 return len(probabilities) - 1  
  
 def initialize\_matrix(self):  
 return [[0.1 if i != j else 0 for j in range(100)] for i in range(100)]  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 problem = TravelingSalesmanProblem()  
 algorithm = AntAlgorithm(problem)  
  
 with open("result.txt", "w") as file:  
 for i in range(50):  
 for j in range(30):  
 algorithm.iterate()  
  
 line = f"{30 \* (i + 1)} {problem.get\_cost(algorithm.get\_solution())}\n"  
 print(line)  
  
 file.write(line)

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

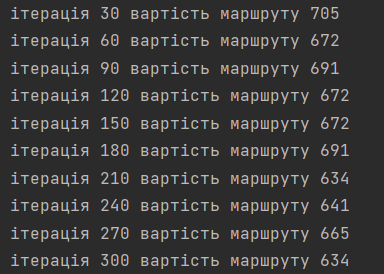


Рисунок 3.1 – номер ітерації та вартість отриманого маршруту

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

|  |  |
| --- | --- |
| **ІТЕРАЦІЯ** | **ЦІЛЬОВА ФУНКЦІЯ** |
| 30 | 650 |
| 60 | 613 |
| 90 | 610 |
| 120 | 610 |
| 150 | 610 |
| 180 | 610 |
| 210 | 610 |
| 240 | 610 |
| 270 | 610 |
| 300 | 648 |
| 330 | 610 |
| 360 | 610 |
| 390 | 613 |
| 420 | 610 |
| 450 | 610 |
| 480 | 610 |
| 510 | 610 |
| 540 | 610 |
| 570 | 613 |
| 600 | 610 |
| 630 | 610 |
| 660 | 610 |
| 690 | 610 |
| 720 | 610 |
| 750 | 610 |
| 780 | 610 |
| 810 | 610 |
| 840 | 610 |
| 870 | 610 |
| 900 | 610 |
| 930 | 610 |
| 960 | 610 |
| 990 | 613 |
| 1020 | 610 |
| 1050 | 610 |
| 1080 | 610 |
| 1110 | 610 |
| 1140 | 610 |
| 1170 | 610 |
| 1200 | 610 |
| 1230 | 610 |
| 1260 | 610 |
| 1290 | 610 |
| 1320 | 610 |
| 1350 | 610 |
| 1380 | 610 |
| 1410 | 610 |
| 1440 | 610 |
| 1470 | 610 |
| 1500 | 610 |

### 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 роботи був розглянутий мурашиний алгоритм, який використовується для вирішення задачі комівояжера. У вигляді програмної реалізації був представлений код на мові Python, і проведений докладний аналіз отриманих результатів.

Згідно з таблицею та графіком, видно, що зі збільшенням кількості ітерацій досягається поліпшення результату. Зменшення необхідного для обходу графа шляху свідчить про ефективність мурашиного алгоритму. Проте важливо відзначити, що з певного моменту спостерігається сповільнення темпів зменшення довжини шляху, або вона може навіть досягти практично сталого значення.

Такий аналіз вказує на те, що, хоча збільшення кількості ітерацій допомагає отримати оптимальніші рішення, існує певний момент, коли додаткові ітерації можуть не призводити до значущого покращення. Це свідчить про важливість балансу між точністю результатів та витратами обчислювальних ресурсів при використанні мурашиного алгоритму для вирішення задачі комівояжера.

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