**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-12 Єльчанінов Артем Юрійович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Сопов О.О.*

Київ 2022

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 numpy as np

class Ant:

def \_\_init\_\_(self, graph\_matrix):

self.graph\_matrix = graph\_matrix

self.position = np.random.randint(len(graph\_matrix))

self.allowed = list(range(len(graph\_matrix)))

self.taboo = [self.position]

self.allowed.remove(self.position)

def restart(self):

self.position = np.random.randint(len(self.graph\_matrix))

self.allowed = list(range(len(self.graph\_matrix)))

self.taboo = [self.position]

self.allowed.remove(self.position)

def add\_pheromone(self):

Q = 10

L = 0

prev = self.taboo[0]

for i in self.taboo[1:]:

L += len(self.graph\_matrix[prev, i])

prev = i

L += len(self.graph\_matrix[self.taboo[0], prev])

pheromone\_add = Q / L

prev = self.taboo[0]

for j in self.taboo[1:]:

self.graph\_matrix[prev, j].pheromone += pheromone\_add

self.graph\_matrix[j, prev].pheromone += pheromone\_add

prev = j

self.graph\_matrix[self.taboo[0], prev].pheromone += pheromone\_add

self.graph\_matrix[prev, self.taboo[0]].pheromone += pheromone\_add

def run(self):

while True:

if len(self.allowed) == 0:

if self.graph\_matrix[self.taboo[0], self.taboo[-1]] == 0:

self.position = np.random.randint(len(self.graph\_matrix))

self.allowed = list(range(len(self.graph\_matrix)))

self.taboo = [self.position]

self.allowed.remove(self.position)

continue

else:

break

elif not self.graph\_matrix[self.position][self.allowed].any():

self.position = np.random.randint(len(self.graph\_matrix))

self.allowed = list(range(len(self.graph\_matrix)))

self.taboo = [self.position]

self.allowed.remove(self.position)

continue

self.step()

def step(self):

r = np.random.rand()

desire\_list = self.want()

for i in range(len(desire\_list)):

if r < desire\_list[i]:

vortex = self.allowed[i]

self.allowed.remove(vortex)

self.taboo.append(vortex)

self.position = vortex

break

else:

r -= desire\_list[i]

def want(self):

desire\_list = []

for index in self.allowed:

desire\_list.append(self.graph\_matrix[self.position][index].desire\_to\_walk())

if len(desire\_list) == 0:

desire\_sum = sum(desire\_list)

else:

desire\_sum = sum(desire\_list)

for i in range(len(desire\_list)):

try:

desire\_list[i] /= desire\_sum

except ZeroDivisionError:

print(desire\_list, self.allowed)

return desire\_list

class Edge:

def \_\_init\_\_(self, length):

self.length = length

self.pheromone = 1

def \_\_len\_\_(self):

return self.length

def \_\_str\_\_(self):

return f"{self.length} {self.pheromone}"

def \_\_repr\_\_(self):

return str(self)

def desire\_to\_walk(self, alfa=2, beta=4):

if self.length == 0:

return 0

return (self.pheromone\*\*alfa) \* ((10/self.length)\*\*beta)

def run\_time(self, p=0.6):

self.pheromone \*= p

class Graph:

def \_\_init\_\_(self, size, min\_edge\_weight=0, max\_edge\_weight=51):

matrix = np.random.randint(min\_edge\_weight, max\_edge\_weight, size=[size, size])

for i in range(len(matrix)):

matrix[i:] = matrix.T[i:]

self.matrix = np.array([[Edge(length) for length in row] for row in matrix])

def run\_time(self):

for row in self.matrix:

for edge in row:

edge.run\_time()

def ant\_algorithm(self, number\_of\_ants):

ants = [Ant(self.matrix) for i in range(number\_of\_ants)]

for i in range(1001):

for r in ants:

r.restart()

for a in ants:

a.run()

for ant in ants:

ant.add\_pheromone()

for row in self.matrix:

for element in row:

element.run\_time()

print(self.get\_best\_len())

return self.get\_best\_len()

def get\_best(self):

allowed = list(range(1, len(self.matrix)))

taboo = [0]

for i in range(len(allowed)):

next\_vortex = max(allowed, key=lambda x: self.matrix[taboo[-1], x].pheromone)

allowed.remove(next\_vortex)

taboo.append(next\_vortex)

return taboo

def get\_best\_len(self):

best\_line = self.get\_best()

L = 0

prev = best\_line[0]

for i in best\_line[1:]:

L += len(self.matrix[prev, i])

prev = i

L += len(self.matrix[best\_line[0], prev])

return L

def main():

g = Graph(250)

g.ant\_algorithm(45)

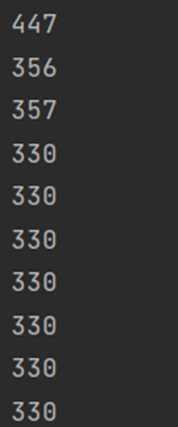
if \_\_name\_\_ == "\_\_main\_\_":

main()

### Приклади роботи

В результаті тестування можна дійти до висновку, що вже після 20-ти ітерацій ми доходимо до досить малих значень (Якщо враховувати, що в нас є 250 вершин, і мінімальна відстань між ними 1, то очевидно, що шлях не може бути коротшим ніж 250, але немає гарантій, що він може мати довжину 250. Щоб це перевірити довелося б перебрати всі 214!/2 варіантів).

На рисунках 3.1 і 3.2 показані приклади роботи програми.



Результат через кожні 20 ітерацій (№1,21,41….)

Рисунок 3.1 – результат через кожні 20 ітерацій)



Результат через кожну ітерацію (№1,2,3,4….)

Результат через кожну ітерацію (№1,2,3,4….)

Рисунок 3.2 – Результат через кожну ітерацію

## Тестування алгоритму

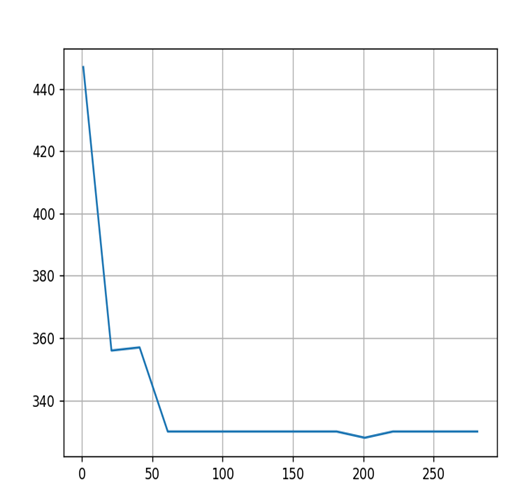
### Значення цільової функції зі збільшенням кількості ітерацій

|  |  |
| --- | --- |
| № ітерації | f(x) |
| 1 | 447 |
| 21 | 356 |
| 41 | 357 |
| 61 | 330 |
| 81 | 330 |
| 101 | 330 |
| 121 | 330 |

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

### Графіки залежності розв'язку від числа ітерацій

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



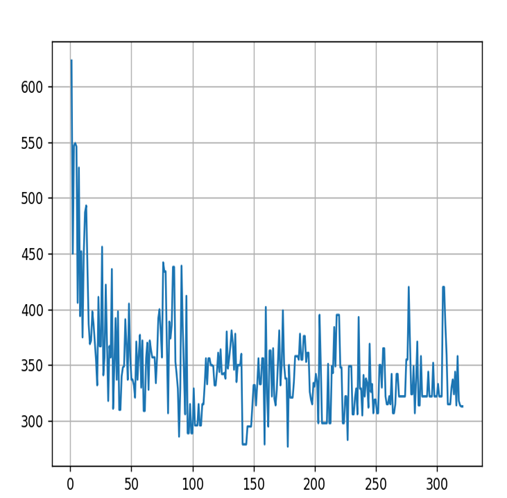


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи вдалося вивчити основні підходи формалізації метаеврестичних алгоритмів і вирішення типових задач з їхньою допомогою, реалізувати мурашиний алгоритм використовуючи мову python та проаналізувати отримані результати

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%.