**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 Васильєв Єгор Костянтинович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

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

Варіант 7

## Програмна реалізація алгоритму

from random import random

from random import randint

from copy import copy

class Genetic:

BACKPACK\_CAPACITY = 150

NUMBER\_OF\_ITEMS = 100

ITERATION\_LIMIT = 1000

PATH\_TO\_FILE = "output.txt"

weight = [randint(1, 5) for \_ in range(NUMBER\_OF\_ITEMS)]

value = [randint(2, 10) for \_ in range(NUMBER\_OF\_ITEMS)]

population = []

probabilities = []

mutation\_probability = 0.05

best = None

best\_ = None

def initialPopulation(self):

for i in range(self.NUMBER\_OF\_ITEMS):

init = [0] \* self.NUMBER\_OF\_ITEMS

init[i] = 1

self.population.append((init, self.fitness(init)))

return True

def fitness(self, chromosome):

sum\_of\_weights = 0

sum\_of\_values = 0

for i in range(len(chromosome)):

if chromosome[i] == 1:

sum\_of\_weights += self.weight[i]

sum\_of\_values += self.value[i]

if sum\_of\_weights > self.BACKPACK\_CAPACITY:

return 0

else:

return sum\_of\_values

def getPossibility(self):

self.probabilities = []

total\_sum = 0

for chromosome in self.population:

total\_sum += chromosome[1]

probability\_sum = 0

for chromosome in self.population:

if total\_sum == 0:

probability\_sum = 0

else:

probability\_sum += float(chromosome[1] / total\_sum)

self.probabilities.append(probability\_sum)

def local\_improvement(self, chromosome):

improved\_chromosome = chromosome.copy()

while True:

i = randint(0, self.NUMBER\_OF\_ITEMS - 1)

if improved\_chromosome[i] == 0:

improved\_chromosome[i] = 1

break

return improved\_chromosome

def nextGeneration(self):

parent\_1 = 0

parent\_2 = -1

r = random()

for parent in range(len(self.probabilities)):

if r < self.probabilities[parent]:

parent\_1 = parent

break

r = random()

for parent in range(len(self.probabilities)):

if r < self.probabilities[parent]:

parent\_2 = parent

break

child\_chromosome = self.crossover(self.population[parent\_1][0], self.population[parent\_2][0])

r = random()

if r < self.mutation\_probability:

mutated\_chromosome = self.mutation(child\_chromosome)

final\_chromosome = self.local\_improvement(mutated\_chromosome)

else:

final\_chromosome = self.local\_improvement(child\_chromosome)

self.population.append((final\_chromosome, self.fitness(final\_chromosome)))

self.population.sort(key=lambda x: x[-1])

self.population.pop(0)

@staticmethod

def mutation(chromosome):

mutated\_chromosome = copy(chromosome)

i1 = randint(0, len(chromosome) - 1)

i2 = randint(0, len(chromosome) - 1)

mutated\_chromosome[i1], mutated\_chromosome[i2] = mutated\_chromosome[i2], mutated\_chromosome[i1]

return mutated\_chromosome

@staticmethod

def crossover(parent\_1, parent\_2):

child\_chromosome = []

for i in range(len(parent\_1)):

if parent\_1[i] != parent\_2[i]:

new\_gene = randint(0, 1)

child\_chromosome.append(new\_gene)

else:

child\_chromosome.append(parent\_1[i])

return child\_chromosome

def write\_info\_to\_file(self, graph\_data):

final\_weight = 0

result = 'Values are written in this format "item number: (item weight, item value)"\n'

for i in range(self.NUMBER\_OF\_ITEMS):

result += f"item {i + 1}: ({self.weight[i]}, {self.value[i]})\n"

result += "---------------------------------\n\n"

result += "Final decision:\n"

for i in range(self.NUMBER\_OF\_ITEMS):

if self.population[-1][0][i] == 1:

result += f"item {i + 1}: ({self.weight[i]}, {self.value[i]})\n"

final\_weight += self.weight[i]

result += "---------------------------------\n\n"

result += "Data for plotting a graph (iteration number, fitness function):\n"

result += graph\_data

with open(self.PATH\_TO\_FILE, "wt") as text\_to\_file:

text\_to\_file.write(result)

def run(self):

graph\_data = ""

print("There are %d item and %d iterations." % (len(self.weight), self.ITERATION\_LIMIT))

print("---------------------------------")

self.initialPopulation()

for i in range(self.ITERATION\_LIMIT):

if (i + 1) % 20 == 0:

graph\_data += f"({i + 1}, {self.population[-1][-1]})\n"

self.getPossibility()

self.nextGeneration()

weight = 0

for i in range(self.NUMBER\_OF\_ITEMS):

if self.population[-1][0][i] == 1:

weight += self.weight[i]

print("Total weight is: ", weight)

print("Total value is: ", self.population[-1][-1])

self.write\_info\_to\_file(graph\_data)

g = Genetic()

g.run()

### Приклади роботи

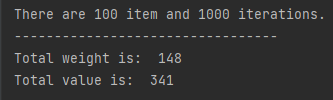
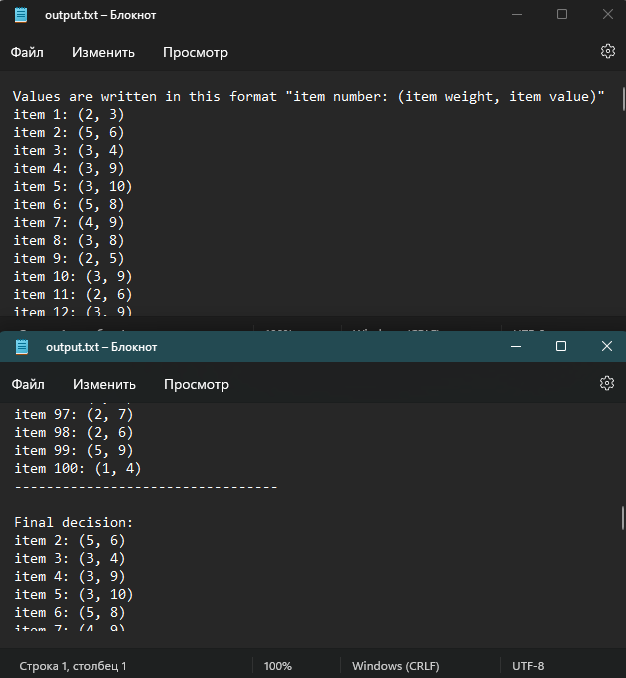
На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 – Вивід кінцевих результатів у консоль

Рисунок 3.2 – Повні дані про розв’язок, записані у текстовий файл

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

Таблиця 3.1 – Характеристика роботи генетичного алгоритму

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість ітерацій | Значення цільової функції | Кількість ітерацій | Значення цільової функції |
| 20 | 32 | 520 | 203 |
| 40 | 46 | 540 | 203 |
| 60 | 46 | 560 | 203 |
| 80 | 63 | 580 | 209 |
| 100 | 63 | 600 | 209 |
| 120 | 83 | 620 | 209 |
| 140 | 83 | 640 | 209 |
| 160 | 83 | 660 | 231 |
| 180 | 86 | 680 | 233 |
| 200 | 97 | 700 | 235 |
| 220 | 107 | 720 | 249 |
| 240 | 107 | 740 | 255 |
| 260 | 107 | 760 | 262 |
| 280 | 118 | 780 | 273 |
| 300 | 125 | 800 | 273 |
| 320 | 136 | 820 | 273 |
| 340 | 144 | 840 | 276 |
| 360 | 144 | 860 | 316 |
| 380 | 144 | 880 | 316 |
| 400 | 153 | 900 | 316 |
| 420 | 154 | 920 | 316 |
| 440 | 166 | 940 | 316 |
| 460 | 166 | 960 | 316 |
| 480 | 181 | 980 | 320 |
| 500 | 181 | 1000 | 341 |

### 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 було розглянуто генетичний алгоритм та досліджено його переваги, серед яких: можливість вирішувати задачу будь-якої розмірності; можливість обмеження рішення задачі, як за часом, так і за заданим значенням критерію; висока швидкодія та можливість використання рішень, отриманих іншими методами; велика кількість налаштувань, які дозволяють змінювати умови і швидкість збіжності. Даний алгоритм було реалізовано на прикладі відомої задачі про рюкзак, використовуючи пропорційну селекцію, рівномірний оператор схрещування, імовірнісний оператор мутації у вигляді зміни місцями двох випадкових генів та локальне покращення. Було побудовано графік залежності значення цільової функції від кількості ітерацій, з якого можна зробити висновок, що завдяки дослідженню одразу декількох «хромосом», цільова функція не впирається в локальний екстремум, а сам алгоритм пропонує на вибір кілька рішень. Загалом алгоритм пропонує задовільні рішення NP-повних задач та підходить для рішення завдань з різними параметрами завдяки великій кількості можливих налаштувань, однак серед недоліків можна відзначити випадковий характер отриманих рішень та незручність представлення задач в термінах генів та хромосом.

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%.