

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)

III-13 Вдовиченко С.Ю.
(шифр, прізвище, ім'я, по батькові)

Перевірив

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

Київ 2022

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
2	ЗАВДАННЯ	4
3	ВИКОНАННЯ.....	10
3.1	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ	10
3.1.1	<i>Вихідний код</i>	<i>10</i>
3.1.2	<i>Приклади роботи</i>	<i>12</i>
3.2	ТЕСТУВАННЯ АЛГОРИТМУ	14
3.2.1	<i>Значення цільової функції зі збільшенням кількості ітерацій .</i>	<i>14</i>
3.2.2	<i>Графіки залежності розв'язку від числа ітерацій</i>	<i>15</i>
	ВИСНОВОК	16
	КРИТЕРІЇ ОЦІНЮВАННЯ	17

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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, L_{\min} 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 ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, L_{\min} 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 ($\alpha = 3$, $\beta = 2$, $\rho = 0,3$, L_{\min} 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 ($\alpha = 2$, $\beta = 4$, ρ

	$= 0,6$, L_{\min} 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 ($\alpha = 4$, $\beta = 2$, $\rho = 0,3$, L_{\min} 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,7$, L_{\min} 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 ($\alpha = 3$, $\beta = 2$, $\rho = 0,7$, L_{min} 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 ($\alpha = 3$, $\beta = 2$, $\rho = 0,6$, L_{min} 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, L_{\min} 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 ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, L_{\min} 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 ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, L_{\min} 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 ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, L_{\min} знайти жадібним алгоритмом, кількість мурах $M = 35$, починають маршрут в різних випадкових вершинах).

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
import random

class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight

class Knapsack:
    def __init__(self, capacity, items):
        self.capacity = capacity
        self.items = items
        self.value = sum([item.value for item in items])
        self.weight = sum([item.weight for item in items])

def generate_items(num_items):
    items = []
    for i in range(num_items):
        value = random.randint(2, 20)
        weight = random.randint(1, 10)
        items.append(Item(value, weight))
    return items

def generate_population(num_individuals, items):
    population = []
    for i in range(num_individuals):
        item = random.choice(items)
        knapsack = Knapsack(200, [item])
        population.append(knapsack)
    return population

def crossover(parent1, parent2):
    crossover_point = random.randint(1, len(parent1.items))
    child_items = parent1.items[:crossover_point] +
parent2.items[crossover_point:]
    return Knapsack(200, child_items)

def mutation(individual, mutation_probability):
    if random.uniform(0, 1) < mutation_probability:
        item_to_remove = random.choice(individual.items)
        individual.items.remove(item_to_remove)
        new_item = random.choice(items)
        individual.items.append(new_item)

def local_improvement(individual, items):
    knapsack_items = individual.items.copy()
    current_weight = individual.weight
    current_value = individual.value

    for item in items:
        if current_weight + item.weight <= 200:
            knapsack_items.append(item)
            current_weight += item.weight
            current_value += item.value
    individual.items = knapsack_items
    individual.weight = current_weight
```

```

        individual.value = current_value

def print_population(population):
    for i, knapsack in enumerate(population):
        print("Knapsack {}: weight = {}, value = {}".format(i+1,
knapsack.weight, knapsack.value))

def print_knapsack(knapsack):
    print("Knapsack : weight = {}, value = {}".format(knapsack.weight,
knapsack.value))

if __name__ == "__main__":
    generations_number = int(input('Enter number of generations: '))
    items = generate_items(100)
    population = generate_population(100, items)

    for generation in range(generations_number):
        new_population = []
        for i in range(100):
            parent1 = random.choice(population)
            parent2 = random.choice(population)
            child = crossover(parent1, parent2)
            mutation(child, 0.1)
            local_improvement(child, items)
            new_population.append(child)
        population = new_population
        total_fitness = sum([knapsack.value for knapsack in population])
        mean_fitness = int(total_fitness / len(population))
        max_fitness = max(population, key=lambda x: x.value)

        best_individual = max(population, key=lambda x: x.value)
        print_knapsack(best_individual)
        print("Best knapsack value:", best_individual.value)
        print("Mean fitness:", mean_fitness)
        print("Max fitness:", max_fitness.value)
        print("Knapsack items: ")
        for item in best_individual.items:
            print(f"Item: value = {item.value},\tweight = {item.weight}")

```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

```
E:\ads_lab4\venv\Scripts\python.exe E:/ads_lab4/main.py
Enter number of generations: 100
Knapsack : weight = 232, value = 635
Best knapsack value: 635
Mean fitness: 577
Max fitness: 635
Knapsack items:
Item: value = 11, weight = 4
Item: value = 19, weight = 8
Item: value = 19, weight = 8
Item: value = 6, weight = 8
Item: value = 7, weight = 8
Item: value = 7, weight = 8
Item: value = 4, weight = 8
Item: value = 15, weight = 4
Item: value = 18, weight = 3
Item: value = 12, weight = 10
Item: value = 18, weight = 6
Item: value = 8, weight = 8
Item: value = 12, weight = 5
Item: value = 16, weight = 9
Item: value = 11, weight = 9
Item: value = 16, weight = 9
Item: value = 16, weight = 9
Item: value = 16, weight = 3
Item: value = 16, weight = 3
```

Рисунок 3.1 – Робота програми для 100 ітерацій

```
E:\ads_lab4\venv\Scripts\python.exe E:/ads_lab4/main.py
Enter number of generations: 1000
Knapsack : weight = 330, value = 837
Best knapsack value: 837
Mean fitness: 718
Max fitness: 837
Knapsack items:
Item: value = 19, weight = 10
Item: value = 19, weight = 10
Item: value = 19, weight = 10
Item: value = 11, weight = 5
Item: value = 11, weight = 5
Item: value = 18, weight = 5
Item: value = 2, weight = 1
Item: value = 7, weight = 1
Item: value = 7, weight = 1
Item: value = 2, weight = 1
Item: value = 5, weight = 5
Item: value = 5, weight = 5
Item: value = 12, weight = 5
Item: value = 11, weight = 1
Item: value = 11, weight = 1
```

Рисунок 3.2 – Робота програми для 1000 ітерацій

3.2 Тестування алгоритму

3.2.1 Значення цільової функції зі збільшенням кількості ітерацій

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

Ітерації	Середнє значення цільової функції
0	11
20	431
40	651
60	465
80	674
100	641
120	634
140	797
160	521
180	787
200	533
260	381
300	592
320	742
420	722
520	677
620	824
720	767
820	621
840	722
920	776
940	624
980	648
1000	761

3.2.2 Графіки залежності розв'язку від числа ітерацій

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

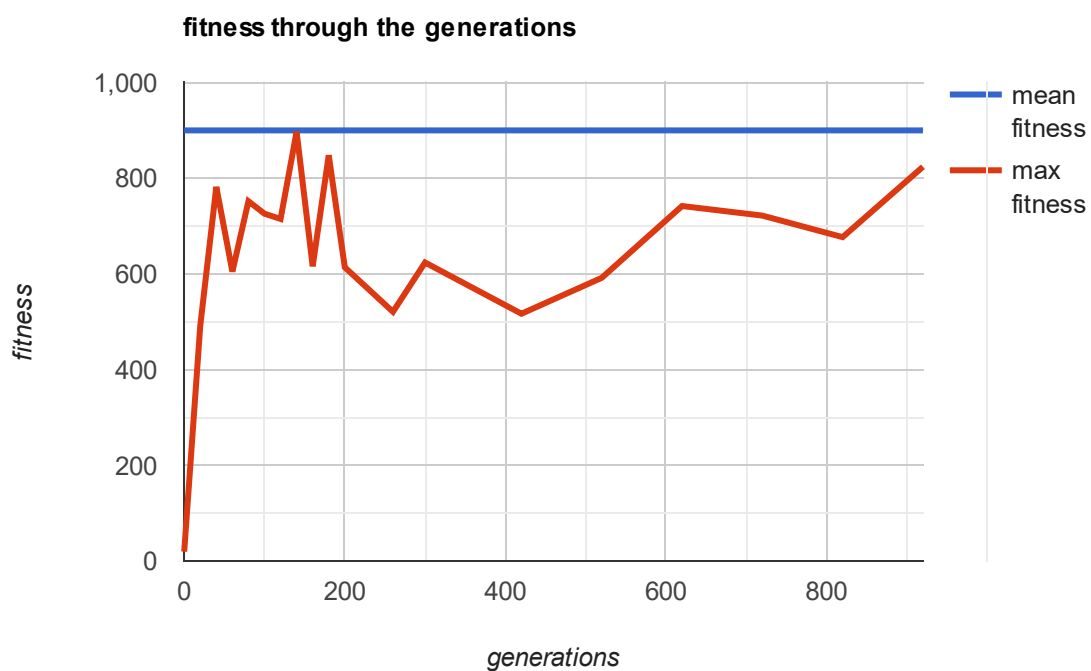


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

ВИСНОВОК

В рамках даної лабораторної роботи я реалізував генетичний алгоритм на прикладі задачі про рюкзак. Генетичний алгоритм в умовах даної задачі складався з вирахування цільової функції, вибірки генів для подальшого кросинговеру, сам кросинговер безпосередньо та можливі мутації генів з ймовірністю 10 відсотків. Суть мутації полягає в тому, що бінарне значення хромосоми буде змінене на протилежне ($1 \Rightarrow 0$ | $0 \Rightarrow 1$). Було проаналізовано алгоритм при різних значеннях вхідних параметрів та побудовано графік залежності значення цільової функції від кількості ітерацій.

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%.