

**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 Нещерет В. О.  
(шифр, прізвище, ім'я, по батькові)

**Перевірив**

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

Київ 2022

## ЗМІСТ

<b>1</b>	<b>МЕТА ЛАБОРАТОРНОЇ РОБОТИ .....</b>	<b>3</b>
<b>2</b>	<b>ЗАВДАННЯ .....</b>	<b>4</b>
<b>3</b>	<b>ВИКОНАННЯ.....</b>	<b>10</b>
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 .....	13
3.2.1	<i>Значення цільової функції зі збільшенням кількості ітерацій .</i>	<i>13</i>
3.2.2	<i>Графіки залежності розв'язку від числа ітерацій .....</i>	<i>14</i>
	<b>ВИСНОВОК .....</b>	<b>15</b>
	<b>КРИТЕРІЇ ОЦІНЮВАННЯ .....</b>	<b>16</b>

## 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 ( $\alpha = 2$ , $\beta = 4$ , $\rho$

	= 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

##### **main.kt**

```
fun main() {
    val numberOfItems = 100
    val populationSize = 100
    val backpackCapacity = 250

    val items = mutableListOf<Item>()
    for (i in 0..numberOfItems) {
        items.add(Item())
    }

    val population = Population(items, populationSize)
    population.createPopulation()

    GeneticAlgorithm(population, backpackCapacity).run()
}
```

##### **GeneticAlgorithm.kt**

```
class GeneticAlgorithm(private val population: Population, private val
backpackCapacity: Int) {
    fun run() {
        for (i in 0..1000) {
            val parent1 = population.getBest()
            var parent2 =
population.individuals[(0..population.populationSize).random()]
            while (parent1 == parent2) {
                parent2 =
population.individuals[(0..population.populationSize).random()]
            }
            val child = parent1.crossover(parent2)
            child.mutate()
            child.localImprovement()
            if (child.weight <= backpackCapacity) {
                population.add(child)
                population.individuals.remove(population.getWorst())
            }
            if (i % 20 == 0) {
                println("$i : ${population.getBest().value}")
            }
        }
        println(population.getBest())
    }
}
```

##### **Population.kt**

```
class Population(val items : MutableList<Item>, val populationSize : Int) {
    val individuals: MutableList<Individual> = mutableListOf()

    fun createPopulation() {
```

```

        for (i in 0..populationSize) {
            individuals.add(Individual(items).also { it.chromosome[i] = true })
        }
    }

    fun add(individual: Individual) {
        individuals.add(individual)
    }

    fun getBest(): Individual {
        individuals.sortBy { it.value }
        return individuals.last()
    }

    fun getWorst(): Individual {
        individuals.sortBy { it.value }
        return individuals.first()
    }
}

```

## Individual.kt

```

class Individual(private val items : MutableList<Item>) {
    val chromosome : MutableList<Boolean> = MutableList(items.size) {false}
    val weight: Int
    get() {
        var sum = 0
        for (i in 0 until items.size) {
            if (chromosome[i]) {
                sum += items[i].weight
            }
        }
        return sum
    }
    val value: Int
    get() {
        var sum = 0
        for (i in 0 until items.size) {
            if (chromosome[i]) {
                sum += items[i].value
            }
        }
        return sum
    }

    fun crossover(other: Individual): Individual {
        val child = Individual(items)
        child.chromosome.clear()
        child.chromosome.addAll(chromosome.subList(0, items.size/4))
        child.chromosome.addAll(other.chromosome.subList(items.size/4,
items.size/2))
        child.chromosome.addAll(chromosome.subList(items.size/2,
items.size/4*3))
        child.chromosome.addAll(other.chromosome.subList(items.size/4*3,
items.size))
        return child
    }

    fun mutate() {
        if (Math.random() < 0.05) {
            val gen1 = (0 until items.size).random()
            val gen2 = (0 until items.size).random()

```

```

        while (gen1 == gen2) {
            gen2 = (0 until items.size).random()
        }
        chromosome[gen1] = chromosome[gen2].also { chromosome[gen2] =
chromosome[gen1] }
    }

    fun localImprovement() {
        var gen = (0 until items.size).random()
        while (chromosome[gen]) {
            gen = (0 until items.size).random()
        }
        chromosome[gen] = true
    }

    override fun toString() : String {
        return "Chromosome = $chromosome\n weight = $weight\n value = $value"
    }
}

```

## Item.kt

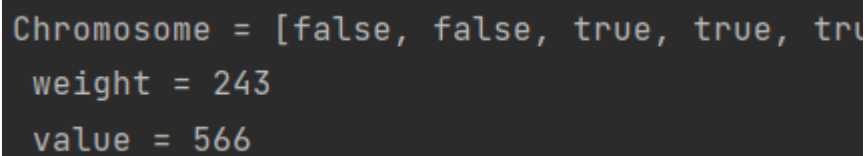
```

class Item(val weight: Int = (1..25).random(),
    val value: Int = (2..30).random())

```

### 3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

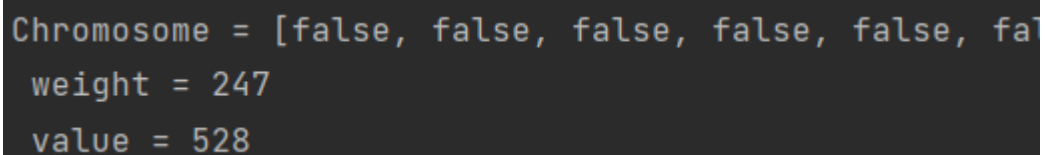


```

Chromosome = [false, false, true, true, true]
weight = 243
value = 566

```

Рисунок 3.1 – Результат роботи алгоритму після 1000 ітерацій



```

Chromosome = [false, false, false, false, false, false]
weight = 247
value = 528

```

Рисунок 3.2 – Результат роботи алгоритму після 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	Цільова функція
0	30
20	72
40	97
60	219
120	254
140	263
180	269
200	320
220	331
240	347
300	357
320	359
340	395
360	418
480	434
580	436
600	453
620	491
740	493
820	506
840	512
920	515
960	526

### 3.2.2 Графіки залежності розв'язку від числа ітерацій

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

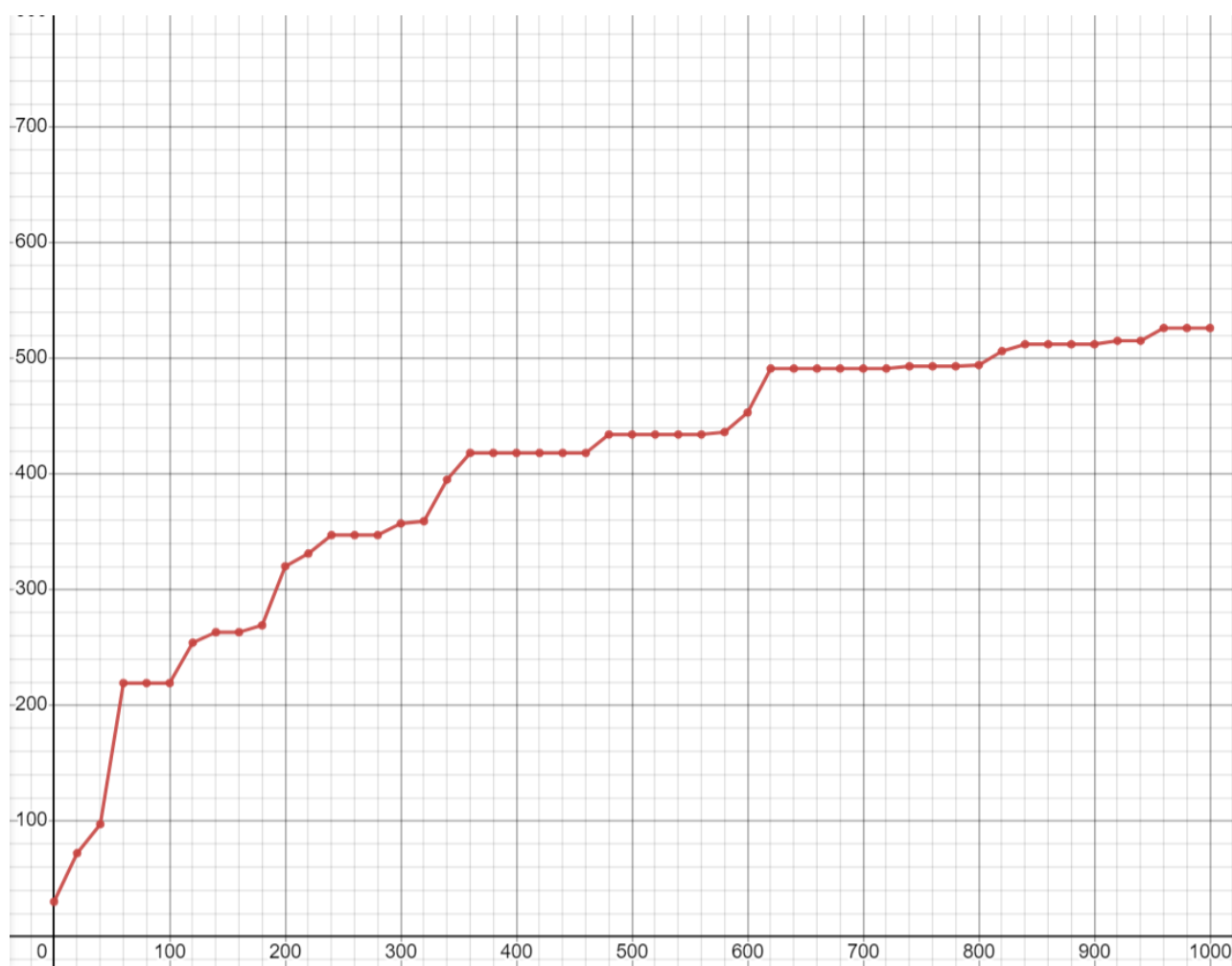


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

## ВИСНОВОК

В рамках даної лабораторної роботи я реалізував генетичний алгоритм на прикладі задачі про рюкзак. Мені довелось розробити власний метод локального покращення, він полягає у тому, що один випадковий ген, що містить негативне значення, може змінити значення на протилежне. Після цього я проаналізував роботу алгоритму та визначив зміну цільової функції в залежності від числа 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%.