

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)

ІІ-14 Качмар Андрій Дмитрович
(шифр, прізвище, ім'я, по батькові)

Перевірив

Головченко М.Н.
(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	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>
	ВИСНОВОК	15
	КРИТЕРІЇ ОЦІНЮВАННЯ	16

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 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
public PopulationNode getPackedKnapsack(Integer iterations) {
    if (capacity <= 0) {
        throw new RuntimeException("Capacity should not be 0 or less");
    }
    if (iterations <= 0) {
        throw new RuntimeException("Iterations should not be 0 or less");
    }
    for (int i = 0; i < iterations; i++) {
        PopulationNode selectedNode = getPopulationSelection();
        PopulationNode afterCross = getPopulationCross(selectedNode, currentRecordNode);

        MutationResponse mutationResponse = getPopulationMutation(afterCross);
        PopulationNode nodeForImprovement = mutationResponse.isSuccessful() ? mutationResponse.getPopulationNode() : afterCross;
        PopulationNode populationNode = getLocalImprovement(nodeForImprovement);

        updateCurrentRecord(populationNode);
        replaceWorstPopulationNode(populationNode);
    }
    return currentRecordNode;
}

public void updateCurrentRecord(PopulationNode populationNode) {
    if (populationNode.getTotalPrice() > currentRecordNode.getTotalPrice() && populationNode.getTotalWeight() <= capacity) {
        currentRecordNode = populationNode;
    }
}
```

```
public void generateInitialPopulation() {
    List<Integer> emptyPopulation = new ArrayList<>(Collections.nCopies(numberOfNodes, 0));
    for (int i = 0; i < numberOfNodes; i++) {
        List<Integer> population = new ArrayList<>(emptyPopulation);
        population.set(i, 1);
        Item currentItem = items.get(i);
        currentPopulation.add(new PopulationNode(population, currentItem.getPrice(), currentItem.getWeight()));
    }
    currentRecordNode = getBestNodeOfPopulation(currentPopulation);
}

public PopulationNode getPopulationSelection() {
    int nodeIndex = getRandomNumber(0, numberOfNodes);
    PopulationNode populationNode = currentPopulation.get(nodeIndex);
    if (populationNode.getNodeId().equals(currentRecordNode.getNodeId())) {
        getPopulationSelection();
    }
    return populationNode;
}
```

```

public PopulationNode getPopulationCross(PopulationNode firstNode, PopulationNode secondNode) {
    List<Integer> selectedItems = new ArrayList<>();
    for (int i = 0; i < numberOfNodes; i++) {
        PopulationNode nodeForInsert = isPopulationCross() ? firstNode : secondNode;
        selectedItems.add(nodeForInsert.getItemsSelection().get(i));
    }
    PopulationNode newNode = new PopulationNode(selectedItems);
    newNode.countParameters(items);
    return newNode;
}

public MutationResponse getPopulationMutation(PopulationNode populationNode) {
    int firstNodeIndex = getRandomNumber(0, numberOfNodes);
    int secondNodeIndex = getRandomNumber(0, numberOfNodes);
    while (secondNodeIndex == firstNodeIndex) {
        secondNodeIndex = getRandomNumber(0, numberOfNodes);
    }
    if (isPerformMutation( mutationProbability: 5) && isEnoughGenes(populationNode)) {
        Integer firstGeneValue = populationNode.getItemsSelection().get(firstNodeIndex);
        Integer secondGeneValue = populationNode.getItemsSelection().get(secondNodeIndex);
        populationNode.selectItem(firstNodeIndex, secondGeneValue);
        populationNode.selectItem(secondNodeIndex, firstGeneValue);
        populationNode.countParameters(items);
    }
    return new MutationResponse(populationNode, isSuccessful: populationNode.getTotalWeight() < capacity);
}

```

```

public PopulationNode getLocalImprovement(PopulationNode populationNode) {
    int dif = getRandomNumber(0, 5);
    int xRange = getRandomNumber(0, numberOfNodes);
    int randomIndex = getRandomNumber(xRange, getRange(dif, xRange));
    populationNode.selectItem(randomIndex, value: 1);
    populationNode.countParameters(items);
    return populationNode;
}

```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

```
Enter backpack capacity: 100  
Solution found  
PopulationNode{vector=[1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,  
0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0], totalPrice=394, totalWeight=158}
```

Рисунок 3.1 – Робота програми при розмірі рюкзака 150 на 1000 ітераціях

[illegible]

Рисунок 3.2 – Робота програми при розмірі рюкзака 500 на 1000 ітераціях

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.

Таблиця 3.1

#	Price	Weight
0	10	1
20	38	11
40	61	22
60	83	35
80	83	35
100	116	44
120	145	60
140	165	72
160	174	80
180	178	72
200	182	71
220	192	76
240	210	86
260	220	96
280	227	102
300	243	116
320	268	119
340	280	126
360	301	133
380	311	136
400	314	134
420	334	143
440	334	143
460	342	143
480	343	146
500	355	149
520	358	150
540	369	149
560	375	150
580	375	150
600	375	150
620	375	150
640	375	150
660	381	149

680	381	149
700	381	149
720	381	149
740	381	149
760	386	150
780	390	149
800	390	149
820	390	149
840	392	149
860	392	149
880	397	150
900	397	150
920	397	150
940	397	150
960	397	150
980	397	150

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 було написано програму для вирішення типової задачі про рюкзак з використанням генетичного алгоритму. Початкова популяція якого складала 100 осіб оператор схрещування рівномірний, мутація з ймовірністю 5%. Було розроблено власний оператор локального покращення який вибирає випадковий предмет з проміжку та вибирає його якщо це можливо. Також було побудовано графік залежності якості розв'язку задачі від числа ітерацій.

Отже виконавши дану лабораторну роботу було вивчено основні підходи формалізації метаевристичних алгоритмів і вирішення типових задач з їхньою допомогою.

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%.