**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-13 Дем’янчук Олександр*

**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

Program.cs

namespace Lab4;  
  
class Program  
{  
 public static void Main(string[] args)  
 {  
 int iterations = Int32.Parse(Console.ReadLine());  
 GeneticAlgorithm ga = new GeneticAlgorithm();  
 ga.Start(iterations);  
 }  
}

RandomExtensions.cs

namespace Lab4;  
  
static class RandomExtensions  
{  
 //Fisher-Yates Algorithm for shuffling an array  
 public static void Shuffle<T> (this Random rng, T[] array)  
 {  
 int n = array.Length;  
 while (n > 1)   
 {  
 int k = rng.Next(n--);  
 (array[n], array[k]) = (array[k], array[n]);  
 }  
 }  
}

Store.cs

namespace Lab4;  
  
public class Store  
{  
 public static int AMT\_OF\_ITEMS = 100;  
 public static int VALUE\_LOWER = 2;  
 public static int VALUE\_UPPER = 10;  
 public static int WEIGHT\_LOWER = 1;  
 public static int WEIGHT\_UPPER = 5;  
 public readonly Tuple<int, int>[] Items;  
  
 public Store()  
 {  
 Items = new Tuple<int, int>[AMT\_OF\_ITEMS];  
 Random rng = new Random();  
 for (int i = 0; i < AMT\_OF\_ITEMS; i++)  
 {  
 Items[i] = new Tuple<int, int>(rng.Next(VALUE\_LOWER, VALUE\_UPPER + 1), rng.Next(WEIGHT\_LOWER, WEIGHT\_UPPER + 1));  
 }  
 }  
}

GeneticAlgorithm.cs

namespace Lab4;  
  
public class GeneticAlgorithm  
{  
 private Store \_store;  
 private bool[][] \_population;  
 private static int \_populationSize = 100;  
 private static int \_chromosomeSize = 100;  
 private static int \_capacity = 150;  
 private static double \_mutationProb = 0.05;  
  
  
 public GeneticAlgorithm()  
 {  
 \_store = new Store();  
 \_population = new bool[\_populationSize][];  
   
 }  
  
 public void Start(int iterations)  
 {  
 SetStartPopulation();  
 int bestItemIndex = GetBestItemIndex();  
 int itr = 0;  
 int best = 0;  
 while (itr < iterations)  
 {  
 itr++;  
 best = 0;  
 int worst = 0;  
 GetExtremeIndices(ref best, ref worst);  
 Random rng = new Random();  
 int secondRandom = best;  
 while (secondRandom == best)  
 secondRandom = rng.Next(0, \_populationSize);  
  
 bool[] successor = UniformCrossover(best, secondRandom);  
 double mutationRand = rng.NextDouble();  
 bool[] mutant = new bool[\_chromosomeSize];  
 if (mutationRand < \_mutationProb)  
 {  
 mutant = Mutate(successor);  
 if(GetTotalWeight(mutant) <= \_capacity)  
 Array.Copy(mutant, successor, \_chromosomeSize);  
 }  
 if(successor[bestItemIndex] == false)   
 successor[bestItemIndex] = !successor[bestItemIndex];  
   
 if(GetTotalWeight(successor) <= \_capacity)  
 Array.Copy(successor, \_population[worst], \_chromosomeSize);  
 }  
 Console.WriteLine("Item store:");  
 Console.Write("Value:\t");  
 for (int i = 0; i < Store.AMT\_OF\_ITEMS; i++)  
 {  
 Console.Write($"{\_store.Items[i].Item1} ");  
 }  
 Console.Write("\nWeight:\t");  
 for (int i = 0; i < Store.AMT\_OF\_ITEMS; i++)  
 {  
 Console.Write($"{\_store.Items[i].Item2} ");  
 }  
  
 PrintResults(iterations, best);  
  
 }  
  
 private void PrintResults(int iterations, int best)  
 {  
 Console.WriteLine($"\nThe best chromosome found in {iterations} iterations: ");  
 int toNum = -1;  
 for (int i = 0; i < \_chromosomeSize; i++)  
 {  
 toNum = (\_population[best][i]) ? 1 : 0;  
 Console.Write($"{toNum} ");   
 }  
  
 int bestValue = 0;  
 int bestWeight = GetTotalWeight(\_population[best]);  
  
 for (int i = 0; i < \_chromosomeSize; i++)  
 {  
 if (\_population[best][i] == true)  
 {  
 bestValue += \_store.Items[i].Item1;  
 }  
 }  
 Console.WriteLine($"\nIt weights {bestWeight} and has value of {bestValue}");  
 }  
  
 private void SetStartPopulation()  
 {  
 int[] randomItems = new int[Store.AMT\_OF\_ITEMS];  
 for (int i = 0; i < \_populationSize; i++)  
 {  
 randomItems[i] = i;  
 \_population[i] = new bool[\_chromosomeSize];  
 for (int j = 0; j < \_chromosomeSize; j++)  
 \_population[i][j] = false;  
 }  
 new Random().Shuffle(randomItems);  
  
 for (int i = 0; i < \_populationSize; i++)  
 {  
 \_population[i][randomItems[i]] = true;  
 }  
 }  
  
 private void GetExtremeIndices(ref int best, ref int worst)  
 {  
 int min = Int32.MaxValue;  
 int max = 0;  
 for (int i = 0; i < \_populationSize; i++)  
 {  
 int totalValue = 0;  
 for (int j = 0; j < \_chromosomeSize; j++)  
 {  
 if(\_population[i][j] == true)  
 {  
 totalValue += \_store.Items[j].Item1;  
 }  
 }  
  
 if (min > totalValue)  
 {  
 min = totalValue;  
 worst = i;  
 }  
  
 if (max < totalValue)  
 {  
 max = totalValue;  
 best = i;  
 }  
 }  
 }  
  
 private bool[] UniformCrossover(int first, int second)  
 {  
 Random rng = new Random();  
 int choice = -1;  
 bool[] crossover = new bool[\_chromosomeSize];  
 for (int i = 0; i < \_chromosomeSize; i++)  
 {  
 choice = rng.Next(0, 2);  
 if (choice == 0)  
 crossover[i] = \_population[first][i];  
 else  
 crossover[i] = \_population[second][i];  
 }  
  
 return crossover;  
 }  
  
 private bool[] Mutate(bool[] successor)  
 {  
 bool[] mutant = new bool[\_chromosomeSize];  
 Array.Copy(successor, mutant, \_chromosomeSize);  
 Random rng = new Random();  
 int first = rng.Next(0, \_chromosomeSize);  
 int second = first;  
 while (second == first)  
 second = rng.Next(0, \_chromosomeSize);  
  
 (mutant[first], mutant[second]) = (mutant[second], mutant[first]);  
 return mutant;  
 }  
  
 private int GetTotalWeight(bool[] chromosome)  
 {  
 int totalWeight = 0;  
 for (int i = 0; i < \_chromosomeSize; i++)  
 {  
 if (chromosome[i] == true)  
 totalWeight += \_store.Items[i].Item2;  
 }  
  
 return totalWeight;  
 }  
  
 private int GetBestItemIndex()  
 {  
 int index = 0;  
 int bestValue = 0;  
 int minWeight = Store.WEIGHT\_UPPER;  
 for (int i = 0; i < Store.AMT\_OF\_ITEMS; i++)  
 {  
 if (minWeight > \_store.Items[i].Item2)  
 minWeight = \_store.Items[i].Item2;  
 }  
  
 for (int i = 0; i < Store.AMT\_OF\_ITEMS; i++)  
 {  
 if (\_store.Items[i].Item2 == minWeight && bestValue < \_store.Items[i].Item1)  
 {  
 bestValue = \_store.Items[i].Item1;  
 index = i;  
 }  
 }  
  
 return index;  
 }  
   
}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 –

A picture containing background pattern

Description automatically generated

Рисунок 3.2 –

A computer screen capture

Description automatically generated with medium confidence

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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.

|  |  |
| --- | --- |
| Кількість ітерацій | Вартість цільової функції |
| 0 | 5 |
| 20 | 55 |
| 40 | 71 |
| 60 | 93 |
| 80 | 101 |
| 100 | 109 |
| 120 | 128 |
| 140 | 135 |
| 160 | 138 |
| 180 | 154 |
| 200 | 176 |
| 220 | 186 |
| 240 | 196 |
| 260 | 205 |
| 280 | 217 |
| 300 | 239 |
| 320 | 264 |
| 340 | 250 |
| 360 | 278 |
| 380 | 282 |
| 400 | 288 |
| 420 | 310 |
| 440 | 306 |
| 460 | 348 |
| 480 | 358 |
| 500 | 371 |
| 520 | 364 |
| 540 | 356 |
| 560 | 394 |
| 580 | 406 |
| 600 | 431 |
| 620 | 456 |
| 640 | 421 |
| 660 | 434 |
| 680 | 453 |
| 700 | 439 |
| 720 | 420 |
| 740 | 408 |
| 760 | 475 |
| 780 | 475 |
| 800 | 439 |
| 820 | 440 |
| 840 | 446 |
| 860 | 429 |
| 880 | 469 |
| 900 | 438 |
| 920 | 412 |
| 940 | 455 |
| 960 | 411 |
| 980 | 424 |
| 1000 | 446 |

Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

Chart, scatter chart

Description automatically generated

Висновок

В рамках даної лабораторної роботи дослідив принцип роботи генетичного алгоритма. В практичній частині роботи реалізував рішення задачі про рюкзак, розробив власний метод локального пошуку, що базується на проходженні по відсутнім предметам у рюкзаку та пошуку серед таких найкращого предмету у відношенні цінність/вага. Дослідив алгоритм на ряді тестів та зобразив на графіку залежність точності результатів, що видає алгоритм, до кількості допустимих 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%.