**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-14 Громов Владислав*

**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

## Програмна реалізація алгоритму

### Вихідний код

**Program.cs**

using static Lab4PA.Funcs;  
  
namespace Lab4PA;  
  
class Program  
{  
 static void Main(String[] args)  
 {  
 Console.WriteLine("Генерация городов...");  
 var cities = Gen(150, 50, 5);  
 Console.WriteLine("Города сгенерировались");  
 var antColony = new AntColony(35, 200, 2, 3, 0.4, 1000, cities);  
 antColony.StartAntColony();  
 }  
}

**Ant.cs**

using static Lab4PA.Funcs;  
  
namespace Lab4PA;  
  
public class Ant  
{  
 public List<KeyValuePair<City, int>> tabooList { get; }  
 private int currentCity;  
 private int startCity;  
 private double[,] pheromones;  
 private City[] cities;  
 private double a, b;  
  
 public Ant(int startCity, City[] cities, double a, double b, double[,] pheromones)  
 {  
 this.startCity = startCity;  
 currentCity = startCity;  
 this.cities = cities;  
 this.a = a;  
 this.b = b;  
 tabooList = new List<KeyValuePair<City, int>>();  
 this.pheromones = pheromones;  
 }  
  
 public void StartTheJourney()  
 {  
 for (int i = 0; i < cities.Length - 1; i++)  
 {  
 var chances = GetChances();  
 var indexOfCity = GetRandomIndex(chances, i);  
 tabooList.Add(new KeyValuePair<City, int>(cities[currentCity], currentCity));  
 currentCity = indexOfCity;  
 }  
 tabooList.Add(new KeyValuePair<City, int>(cities[currentCity], currentCity));  
 tabooList.Add(new KeyValuePair<City, int>(cities[startCity], startCity));  
 }  
  
 private double[] GetDesires()  
 {  
 var desires = new double[cities.Length];  
 for (var i = 0; i < cities.Length; i++)  
 {  
 if (i == currentCity || IsKeyPairListContains(tabooList, i))  
 {  
 desires[i] = 0;  
 continue;  
 }  
  
 var desire = Pow(pheromones[currentCity, i] \* 100000, a)  
 \* Pow( 30.0 / (cities[currentCity] - cities[i]), b);  
   
 if (desire == 0)  
 desires[i] = double.**Epsilon**;  
 else  
 desires[i] = desire;  
  
 }  
  
 return desires;  
 }  
  
   
  
 private static bool IsKeyPairListContains(List<KeyValuePair<City, int>> list, int i)  
 {  
 return list.Any(keyValuePair => keyValuePair.Value == i);  
 }  
   
  
 private double[] GetChances()  
 {  
 var desires = GetDesires();  
 var chances = new double[desires.Length];  
  
 var sumOfDesires = desires.Sum();  
  
 for (var i = 0; i < desires.Length; i++)  
 {  
 var ss = desires[i] \* 100 / sumOfDesires;  
 chances[i] = ss;  
 }  
   
  
 return chances;  
 }  
  
   
 private int GetRandomIndex (double[] proportions, int k)  
 {  
 var proportionsSorted = new double[proportions.Length];  
 Array.Copy(proportions, proportionsSorted, proportions.Length);  
 Array.Sort(proportionsSorted);  
 var randomValue = new Random().NextDouble() \* 100;  
 int indexOfNum = -1;  
 for (int i = 0; i < proportionsSorted.Length; i++)  
 {  
 if (randomValue > proportionsSorted[i])  
 randomValue -= proportionsSorted[i];  
 else  
 {  
 indexOfNum = i;  
 break;  
 }  
 }  
  
 if (indexOfNum == -1) throw new Exception("All chances == 0");  
  
 for (int i = 0; i < proportions.Length; i++)  
 {  
 if (proportions[i] == proportionsSorted[indexOfNum]) return i;  
 }  
  
 throw new Exception("All chances == 0");  
 }  
}

**AntColony.cs**

namespace Lab4PA;  
  
public class AntColony  
{  
 private int ants;  
 private int iterations;  
 private double p, Q, a, b;  
 private City[] cities;  
  
 public AntColony(int ants, int iterations, double a, double b, double p, double q, City[] cities)  
 {  
 this.ants = ants;  
 this.iterations = iterations;  
 this.p = 1 - p;  
 Q = q;  
 this.a = a;  
 this.b = b;  
 this.cities = cities;  
 }  
  
 public double StartAntColony()  
 {  
   
 var rnd = new Random();  
 var pheromones = CreatePheromoneMatrix(cities.Length);  
   
 double distance = 0;  
 var minDistance = double.**MaxValue**;  
 List<KeyValuePair<City, int>> minJourney = null;  
 for (int i = 0; i < iterations; i++)  
 {  
 var pheromonesCopy = MatrixCopy(pheromones);  
 pheromones = EvaporationOfPheromones(pheromones);  
 for (int j = 0; j < ants; j++)  
 {  
   
 var ant = new Ant(rnd.Next(0, cities.Length), cities, a, b, pheromonesCopy);  
 ant.StartTheJourney();  
 var antJourney = ant.tabooList;  
 distance = DistanceOfTheJourney(antJourney);  
 if (distance < minDistance)  
 {  
 minDistance = distance;  
 minJourney = antJourney;  
 }  
 pheromones = SetNewPheromones(antJourney, distance, pheromones);  
 }  
  
 Console.WriteLine($"Дистанция после {i+1} итерации: {distance}");  
  
 }  
  
 if (minJourney != null)  
 Console.WriteLine($"Лучший путь:\n{JourneyToString(minJourney)}\nДистанция: {minDistance}");  
 return minDistance;  
 }  
  
 private string JourneyToString(List<KeyValuePair<City, int>> antJourney)  
 {  
 var citiesIndex = antJourney.Select(el => el.Value).ToList();  
 return string.Join('-', citiesIndex);  
 }  
 private double[,] SetNewPheromones(List<KeyValuePair<City, int>> antJourney, double distance, double[,] pheromones)  
 {  
 for (int i = 0; i < antJourney.Count - 1; i++)  
 pheromones = ChangeValueOfPheromones(pheromones, antJourney[i].Value, antJourney[i + 1].Value, Q / distance);  
   
 return pheromones;  
 }  
  
 private double DistanceOfTheJourney(List<KeyValuePair<City, int>> antJourney)  
 {  
 var distance = 0.0;  
 for (var i = 0; i < antJourney.Count-1; i++)  
 distance += antJourney[i].Key - antJourney[i + 1].Key;  
   
 return distance;  
 }  
   
 private double[,] EvaporationOfPheromones(double[,] pheromones)  
 {  
 for (int i = 0; i < pheromones.GetLength(0); i++)  
 {  
 for (int j = 0; j < pheromones.GetLength(1); j++)  
 {  
 pheromones[i, j] \*= p;  
 }  
 }  
  
 return pheromones;  
 }  
  
 private double[,] MatrixCopy(double[,] matrix)  
 {  
 var copy = new double[matrix.GetLength(0), matrix.GetLength(1)];  
 for (int i = 0; i < copy.GetLength(0); i++)  
 {  
 for (int j = 0; j < copy.GetLength(1); j++)  
 {  
 copy[i, j] = matrix[i, j];  
 }  
 }  
 return copy;  
 }  
   
 private double[,] ChangeValueOfPheromones(double[,] pheromones, int i, int j, double value)  
 {  
 pheromones[i, j] = value;  
 pheromones[j, i] = value;  
 return pheromones;  
 }  
   
 private double[,] CreatePheromoneMatrix(int numOfCities)  
 {  
 var pheromone = new double[numOfCities, numOfCities];  
 for (int i = 0; i < pheromone.GetLength(0); i++)  
 {  
 for (int j = 0; j < pheromone.GetLength(1); j++)  
 {  
 pheromone[i, j] = 2;  
 }  
 }  
  
 return pheromone;  
 }  
}

**City.cs**

using static Lab4PA.Funcs;  
  
namespace Lab4PA;  
  
public class City  
{  
 private double X;  
 private double Y;  
  
 public City(double x, double y)  
 {  
 X = x;  
 Y = y;  
 }  
   
 public static double operator -(City c1, City c2)  
 {  
 return Math.Sqrt(Pow(c1.X - c2.X, 2) + Pow(c1.Y - c2.Y, 2));  
 }  
  
 public override string ToString()  
 {  
 return $"({X} ; {Y})";  
 }  
}

**Funcs.cs**

namespace Lab4PA;  
  
public static class Funcs  
{  
  
 public static double Pow(double x, double pow)  
 {  
 double c = 1;  
 for (int i = 0; i < pow; i++)  
 c \*= x;  
  
 return c;  
 }  
  
 public static City[] Gen(int numOfCities, int maxDistance, int minDistance)  
 {  
  
 var ss = new List<City>();  
 var rnd = new Random();  
 while (ss.Count < numOfCities)  
 {  
 var c = new City(rnd.NextDouble() \* maxDistance \* 2, rnd.NextDouble() \* maxDistance \* 2);  
 var f = true;  
 foreach (var p in ss)  
 {  
 if (c - p < minDistance )  
 {  
 f = false;  
 break;  
 }  
 }  
 if (f) ss.Add(c);  
 }  
  
 return ss.ToArray();  
 }  
   
}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 –

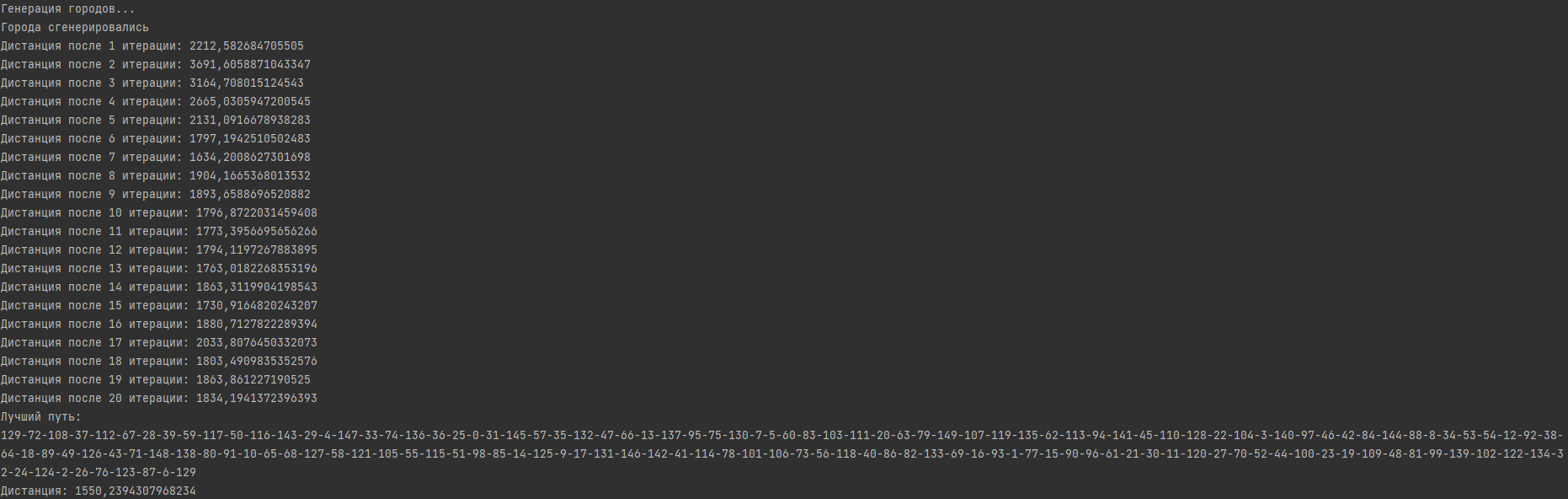
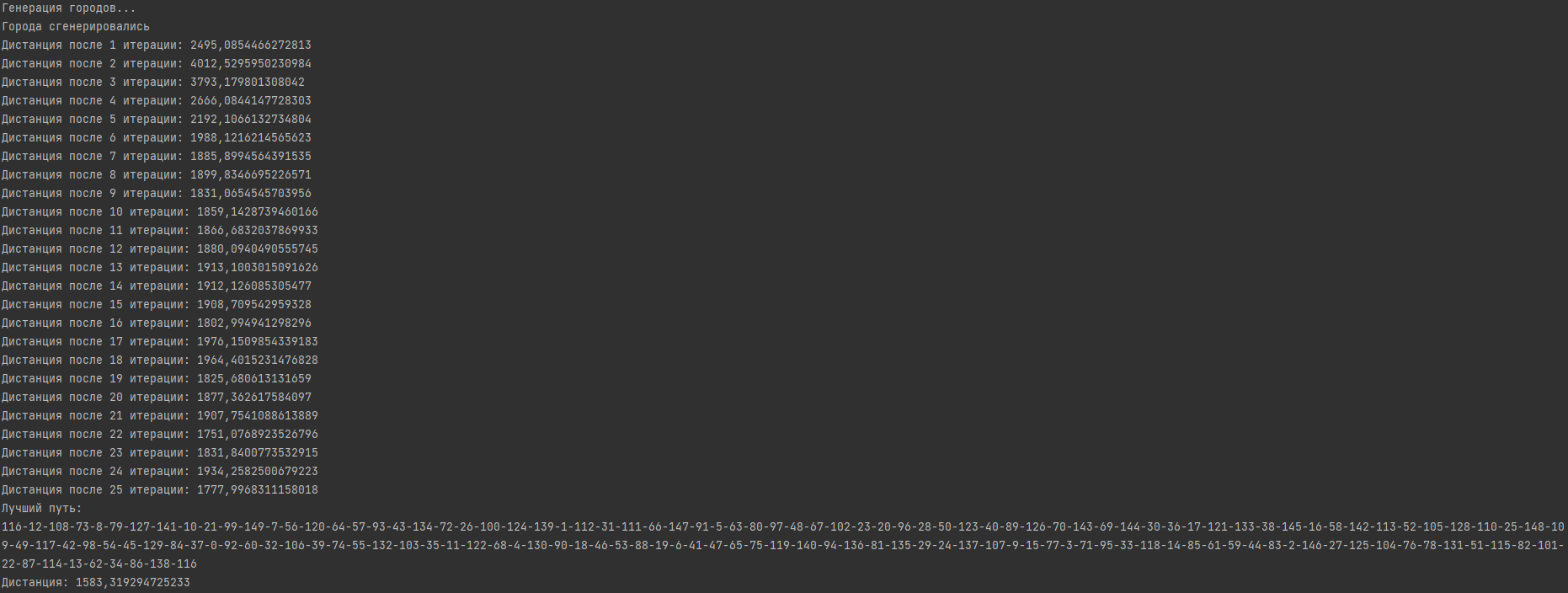


Рисунок 3.2 – 

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

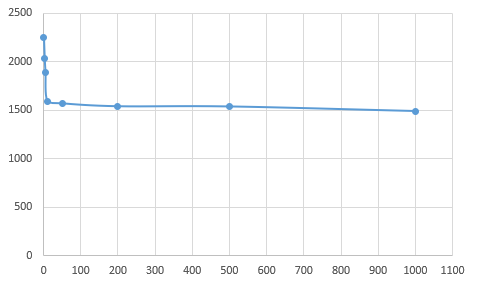
У таблиці 3.1 наведено значення цільової функції зі збільшенням кількості ітерацій.

|  |  |
| --- | --- |
| 1 | 2245,80 |
| 3 | 2036,68 |
| 5 | 1892,29 |
| 10 | 1593,27 |
| 50 | 1565,83 |
| 200 | 1535,62 |
| 500 | 1533,45 |
| 1000 | 1485,97 |

### 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 досліджував мурашиний алгоритм, а також реалізував його, що дало мені багато практичних навичок.

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%.