**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 Крупосій Вадим Сергійович*

**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

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab3ANT

{

internal class Program

{

static void Main(string[] args)

{

Graph graph = new Graph(250);

GreedySearch greedySearch = new GreedySearch(graph);

int lMin = greedySearch.GetMinLenght();

Console.WriteLine($"lMin: {lMin}");

double[,] pheromone = new double[graph.Matrix.GetLength(0), graph.Matrix.GetLength(1)];

for(int i = 0; i < pheromone.GetLength(0); i++)

{

for(int j = 0; j < pheromone.GetLength(1); j++)

{

pheromone[i, j] = 0.0001;

}

}

ACO aco = new ACO(graph.Matrix, pheromone, lMin);

aco.Calculate();

}

}

}

ACO.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab3ANT

{

public class ACO

{

public double Alpha { get; set; }

public double Beta { get; set; }

public double Ro { get; set; }

private int \_lMin;

private int[,] \_distancesMatrix;

private double[,] \_pheromoneMatrix;

private int \_countOfAnt;

private int \_tMax;

public ACO(int[,] distanceMatrix, double[,] pheromoneMatrix, int lMin)

{

\_distancesMatrix = distanceMatrix;

\_pheromoneMatrix = pheromoneMatrix;

Alpha = 4;

Beta = 2;

Ro = 0.3;

\_lMin = lMin;

\_countOfAnt = 45;

\_tMax = 10;

}

public List<Ant> GetAnts()

{

List<Ant> ants = new List<Ant>();

Random rnd = new Random();

for (int i = 0; i < \_countOfAnt; i++)

{

int randomPosition = 0;

do

{

randomPosition = rnd.Next(0, \_distancesMatrix.GetLength(0));

} while (ants.Where(x => x.Position == randomPosition).Count() != 0);

ants.Add(new Ant(randomPosition));

}

return ants;

}

public int GetMaxIndex(double[] array)

{

double max = array.Max();

for (int i = 0; i < array.Length; i++)

{

if (max == array[i]) return i;

}

return -1;

}

public int GetNewPosition(int currPos, List<int> visitedPos)

{

double[] prob = new double[\_distancesMatrix.GetLength(0)];

for (int i = 0; i < prob.Length; i++)

{

if (visitedPos.Contains(i))

{

prob[i] = 0;

}

else

{

prob[i] = Math.Pow(\_pheromoneMatrix[currPos, i], Alpha) \* (Math.Pow((1d / \_distancesMatrix[currPos, i]), Beta));

}

}

Random rnd = new Random();

double newPos = rnd.NextDouble() \* prob.ToList().Sum();

double sum = 0;

int result = GetMaxIndex(prob);

while (sum < newPos)

{

result = GetMaxIndex(prob);

sum += prob[result];

prob[result] = 0;

}

return result;

}

public int GetLength(List<int> way)

{

int length = 0;

for (int i = 0; i < way.Count - 1; i++)

{

length += \_distancesMatrix[way[i], way[i + 1]];

}

return length + \_distancesMatrix[\_distancesMatrix.GetLength(0) - 1, 0];

}

public double[,] GetPheromonic(List<List<int>> ways)

{

for (int i = 0; i < \_pheromoneMatrix.GetLength(0); i++)

{

for (int j = 0; j < \_pheromoneMatrix.GetLength(1); j++)

{

\_pheromoneMatrix[i, j] \*= (1d - Ro);

}

}

foreach (var way in ways)

{

int delta = \_lMin / GetLength(way);

for (int i = 0; i < way.Count - 1; i++)

{

\_pheromoneMatrix[way[i], way[i + 1]] += delta;

\_pheromoneMatrix[way[i + 1], way[i]] += delta;

}

\_pheromoneMatrix[way[ways.Count - 1], way[0]] += delta;

\_pheromoneMatrix[way[0], way[ways.Count - 1]] += delta;

}

return \_pheromoneMatrix;

}

public void Calculate()

{

List<Ant> ants = GetAnts();

int bestLength = int.MaxValue;

List<int> bestWay = new List<int>();

for (int i = 0; i < \_tMax; i++)

{

List<List<int>> ways = new List<List<int>>();

for (int j = 0; j < ants.Count; j++)

{

int currPos = ants[j].Position;

List<int> visPos = new List<int>();

while (visPos.Count < \_distancesMatrix.GetLength(0))

{

visPos.Add(currPos);

currPos = GetNewPosition(currPos, visPos);

}

List<int> tWay = new List<int>();

for (int k = 0; k < visPos.Count; k++)

{

tWay.Add(visPos[k]);

}

ways.Add(tWay);

}

foreach(var way in ways)

{

int length = GetLength(way);

if(length < bestLength)

{

bestWay = way;

bestLength = length;

}

}

\_pheromoneMatrix = GetPheromonic(ways);

}

Console.WriteLine($"Best way: ");

foreach(var w in bestWay)

{

Console.Write(w + " ");

}

Console.WriteLine($"\nBest length: {bestLength}");

}

}

}

Ant.cs  
namespace Lab3ANT

{

public class Ant

{

public int Position { get; set; }

public Ant(int position)

{

Position = position;

}

}

}

Graph.cs

using System;

using System.Runtime.Serialization.Formatters;

namespace Lab3ANT

{

public class Graph

{

public int MinDistance { get; } = 1;

public int MaxDistance { get; } = 40;

public int[,] Matrix { get; set; }

public Graph(int numberOfVertexes)

{

Matrix = new int[numberOfVertexes, numberOfVertexes];

Random rnd = new Random();

for (int i = 0; i < Matrix.GetLength(0); i++)

{

for (int j = 0; j < Matrix.GetLength(1); j++)

{

if (i == j)

{

Matrix[i, j] = 0;

}

else

{

Matrix[i, j] = Matrix[j, i] = rnd.Next(MinDistance, MaxDistance + 1);

}

}

}

}

}

}

Greedysearch.cs

using System.Linq;

using System.Collections.Generic;

namespace Lab3ANT

{

public class GreedySearch

{

public Graph Graph { get; set; }

public GreedySearch(Graph graph)

{

this.Graph = graph;

}

public int GetMin(List<int> revVert, int[] distance)

{

int min = Graph.MaxDistance;

int minI = 0;

for (int i = 0; i < distance.Length; i++)

{

if (distance[i] < min && !revVert.Contains(i) && distance[i] > 0)

{

min = distance[i];

minI = i;

}

}

return minI;

}

public int GetMinLenght()

{

int result = 0;

int currVert = 0;

List<int> revVert = new List<int>();

while (revVert.Count < Graph.Matrix.GetLength(0) - 1)

{

List<int> temp = new List<int>();

for (int i = 0; i < Graph.Matrix.GetLength(1); i++)

{

temp.Add(Graph.Matrix[currVert, i]);

}

int newVert = GetMin(temp, revVert.ToArray());

result += Graph.Matrix[newVert, currVert];

revVert.Add(currVert);

currVert = newVert;

}

result += Graph.Matrix[currVert, 0];

revVert.Add(currVert);

return result;

}

}

}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 –

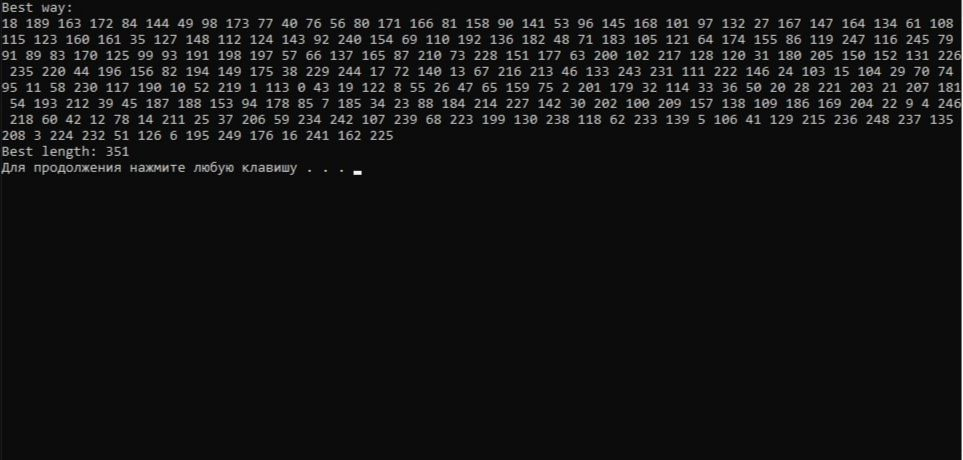


Рисунок 3.2 –

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість ітерацій | Тест 1 | Тест 2 | Середнє значення |
| 0 | 601 | 526 | 563,5 |
| 20 | 581 | 511 | 546 |
| 40 | 562 | 511 | 536,5 |
| 60 | 538 | 511 | 524,5 |
| 80 | 538 | 511 | 524,5 |
| 100 | 538 | 495 | 516,5 |
| 120 | 538 | 495 | 516,5 |
| 140 | 538 | 495 | 516,5 |
| 160 | 513 | 495 | 504 |
| 180 | 513 | 495 | 504 |
| 200 | 291 | 226 | 258,5 |
| 220 | 285 | 226 | 255,5 |
| 240 | 278 | 226 | 252 |
| 260 | 278 | 226 | 252 |
| 280 | 278 | 226 | 252 |
| 320 | 278 | 226 | 252 |
| 360 | 278 | 226 | 252 |
| 400 | 278 | 226 | 252 |
| 440 | 278 | 226 | 252 |
| 480 | 278 | 226 | 252 |
| 520 | 278 | 226 | 252 |
| 560 | 278 | 226 | 252 |
| 600 | 278 | 226 | 252 |
| 640 | 278 | 226 | 252 |
| 680 | 278 | 226 | 252 |
| 720 | 278 | 226 | 252 |
| 760 | 278 | 226 | 252 |
| 800 | 278 | 226 | 252 |
| 840 | 278 | 226 | 252 |
| 880 | 278 | 226 | 252 |
| 920 | 278 | 226 | 252 |
| 960 | 276 | 226 | 251 |
| 1000 | 276 | 226 | 251 |

### Графіки залежності розв'язку від числа ітерацій

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

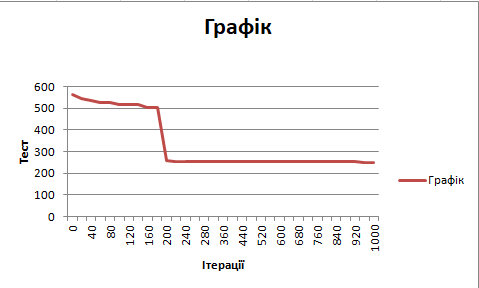


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи я вивчив різноманітні алгоритми оптимізації (метаевристичні алгоритми), навчився розв’язувати базові задачі з їх допомогою, реалізував мурашиний алгоритм (ACO) та провів його аналіз.

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%.