**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**”





Київ 2022

Зміст

[**1**](#_heading=h.30j0zll) **Мета лабораторної роботи 3**

[**2**](#_heading=h.1fob9te) **Завдання 4**

[**3**](#_heading=h.3znysh7) **Виконання 10**

[3.1](#_heading=h.2et92p0) Програмна реалізація алгоритму 10

[*3.1.1*](#_heading=h.tyjcwt) *Вихідний код 10*

[*3.1.2*](#_heading=h.3dy6vkm) *Приклади роботи 10*

[3.2](#_heading=h.1t3h5sf) Тестування алгоритму 11

[*3.2.1*](#_heading=h.4d34og8) *Значення цільової функції зі збільшенням кількості ітерацій 11*

[*3.2.2*](#_heading=h.2s8eyo1) *Графіки залежності розв'язку від числа ітерацій 11*

[**Висновок 12**](#_heading=h.17dp8vu)

[**Критерії оцінювання 13**](#_heading=h.3rdcrjn)

# 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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Salesman\_ant\_algorithm

{

internal class Program

{

static void Main(string[] args)

{

AntColonyAlgorithm colonyAlgorithm = new AntColonyAlgorithm();

colonyAlgorithm.AntAlgorithm();

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Salesman\_ant\_algorithm

{

class Node

{

public int num;

public double[] edges = new double[200];

public double[] pheromones = new double[200];

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Salesman\_ant\_algorithm

{

class Ant

{

public List<Node> proccessedNodes = new List<Node>();

public List<Node> unproccessedNodes = new List<Node>();

public Node position;

public int pheromone = 5;

public Ant(Node startingPosition, List<Node> allNodes)

{

position = startingPosition;

proccessedNodes.Add(startingPosition);

unproccessedNodes = CopyList(allNodes);

unproccessedNodes.Remove(position);

}

public void Move(int A, int B)

{

List<double> probs = new List<double>();

List<double> nums = new List<double>();

for (int i = 0; i < unproccessedNodes.Count; i++)

{

double num = Math.Pow(proccessedNodes[proccessedNodes.Count - 1].pheromones[unproccessedNodes[i].num], A) \*

Math.Pow(1 / proccessedNodes[proccessedNodes.Count - 1].edges[unproccessedNodes[i].num], B);

nums.Add(num);

}

for (int i = 0; i < nums.Count; i++)

probs.Add(nums[i] / Sum(nums));

Random rnd = new Random();

double random = rnd.NextDouble();

for (int i = 0; i < probs.Count; i++)

{

if (random < Sum(probs.GetRange(0, i + 1)))

{

position = unproccessedNodes[i];

unproccessedNodes.Remove(position);

proccessedNodes.Add(position);

i = 1000;

}

}

}

public double GetLength()

{

double length = 0;

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

length += proccessedNodes[i].edges[proccessedNodes[i + 1].num];

length += proccessedNodes[proccessedNodes.Count - 1].edges[proccessedNodes[0].num];

return length;

}

public double Sum(List<double> nums)

{

double sum = 0;

foreach (var num in nums)

sum += num;

return sum;

}

public void LeavePheromone()

{

double l = GetLength();

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

{

proccessedNodes[i].pheromones[proccessedNodes[i + 1].num] += AntColonyAlgorithm.Lmin / l;

proccessedNodes[i + 1].pheromones[proccessedNodes[i].num] = proccessedNodes[i].pheromones[proccessedNodes[i + 1].num];

}

proccessedNodes[proccessedNodes.Count - 1].pheromones[proccessedNodes[0].num] += AntColonyAlgorithm.Lmin / l;

proccessedNodes[0].pheromones[proccessedNodes[proccessedNodes.Count - 1].num] =

proccessedNodes[proccessedNodes.Count - 1].pheromones[proccessedNodes[0].num];

}

public List<Node> CopyList(List<Node> baseList)

{

List<Node> newList = new List<Node>();

foreach (var element in baseList)

newList.Add(element);

return newList;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace Salesman\_ant\_algorithm

{

class AntColonyAlgorithm

{

public int alpha = 3;

public int beta = 2;

public float rho = 0.7f;

public const int edgeNumber = 200;

public int edgeMin = 1;

public int edgeMax = 40;

public int antQty = 45;

public static double Lmin;

private Node[] nodes = new Node[edgeNumber];

public void AntAlgorithm()

{

PlaceNodes();

Lmin = SeekLMin();

Random rand = new Random();

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

for (int i = 0; i < 100; i++)

{

ants.Clear();

for (int j = 0; j < antQty; j++)

ants.Add(new Ant(nodes[rand.Next(0, edgeNumber-1)], nodes.ToList()));

for (int j = 0; j < edgeNumber-1; j++)

{

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

ants[k].Move(alpha, beta);

}

for (int j = 0; j < edgeNumber; j++)

{

for (int k = 0; k < edgeNumber - 1; k++)

nodes[j].pheromones[k] \*= 1 - rho;

}

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

ants[k].LeavePheromone();

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

ants[k].Move(alpha, beta);

if (i%20 == 0)

{

double curTrail = 0;

double minTrail = double.MaxValue;

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

{

curTrail = ants[j].GetLength();

if (curTrail < minTrail)

minTrail = curTrail;

}

Console.WriteLine(i + "," + minTrail);

}

}

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

{

string path = "";

for (int j = 0; j < ants[i].proccessedNodes.Count; j++)

path += ants[i].proccessedNodes[j].num.ToString() + " - ";

Console.WriteLine(path + "\n");

}

}

public void PlaceNodes()

{

Random rnd = new Random();

for (int i = 0; i < edgeNumber; i++)

{

nodes[i] = new Node();

nodes[i].num = i;

}

for (int i = 0; i < edgeNumber; i++)

{

for (int j = 0; j < edgeNumber; j++)

{

nodes[i].pheromones[j] = 0.5f;

if (nodes[i].num == j)

nodes[i].edges[j] = 0;

else if (nodes[j].edges[i] == 0)

{

float number = rnd.Next(edgeMin, edgeMax);

nodes[i].edges[j] = number;

nodes[j].edges[i] = number;

}

else if (nodes[j].edges[i] != 0)

nodes[i].edges[j] = nodes[j].edges[i];

}

}

}

public double SeekLMin()

{

Ant ant = new Ant(nodes[0], nodes.ToList());

for (int i = 0; i < edgeNumber - 1; i++)

ant.Move(0, 1);

return ant.GetLength();

}

}

}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 –

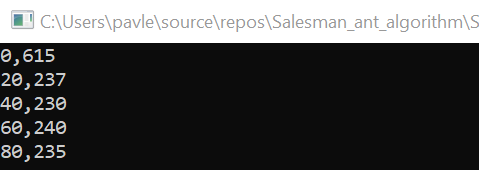
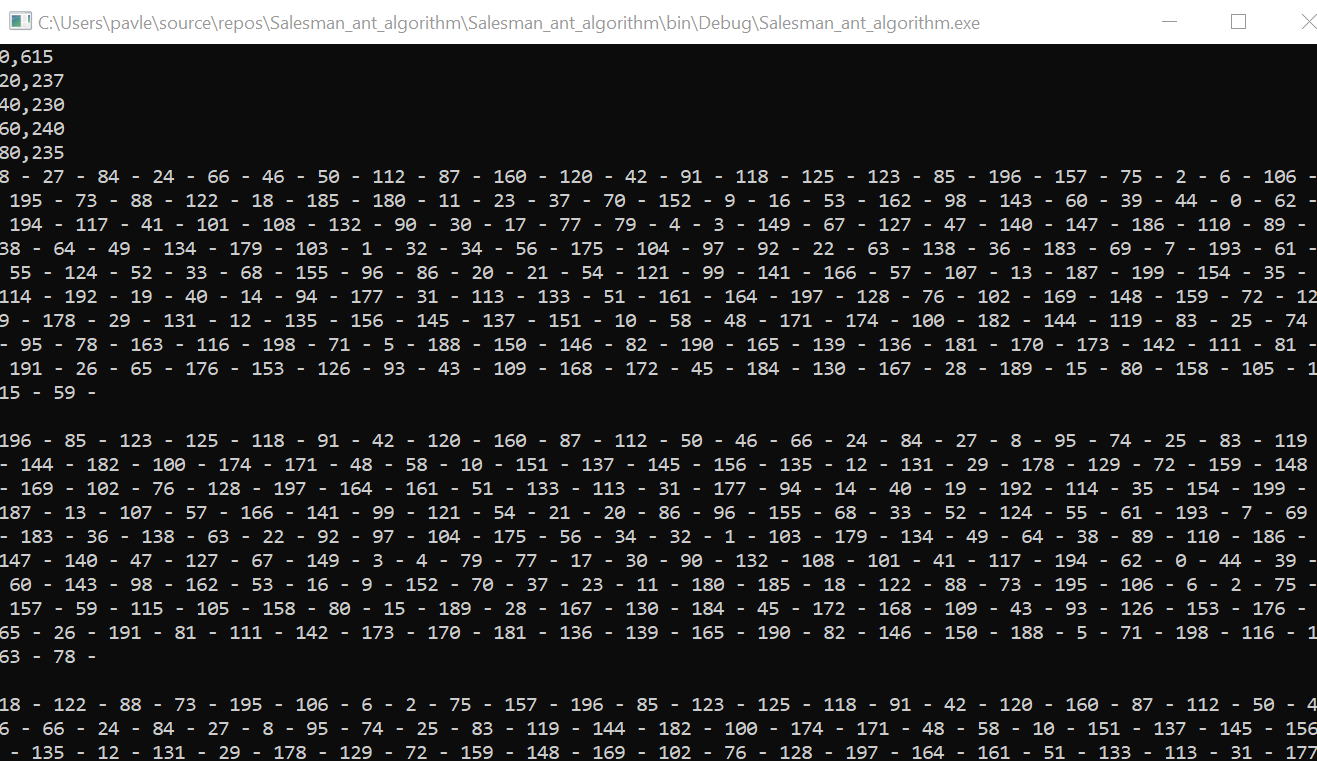


Рисунок 3.2 –



## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

Таблиця 3.1

| 20 | 238 |
| --- | --- |
| 40 | 255 |
| 60 | 261 |
| 80 | 252 |
| 100 | 259 |
| 120 | 262 |
| 140 | 262 |
| 160 | 266 |
| 180 | 258 |
| 200 | 258 |
| 220 | 266 |
| 240 | 27 |
| 260 | 265 |
| 280 | 252 |
| 300 | 243 |
| 320 | 252 |
| 340 | 27 |
| 360 | 243 |
| 380 | 262 |
| 400 | 268 |

Продовження таблиці 3.1

| 420 | 246 |
| --- | --- |
| 440 | 262 |
| 460 | 262 |
| 480 | 266 |
| 500 | 267 |
| 520 | 266 |
| 540 | 259 |
| 560 | 272 |
| 580 | 25 |
| 600 | 266 |
| 620 | 264 |
| 640 | 273 |
| 660 | 27 |
| 680 | 246 |
| 700 | 246 |
| 720 | 28 |
| 740 | 272 |
| 760 | 27 |
| 780 | 26 |
| 800 | 229 |
| 820 | 24 |
| 840 | 24 |
| 860 | 265 |
| 880 | 247 |

Продовження таблиці 3.1

| 900 | 262 |
| --- | --- |
| 920 | 251 |
| 940 | 267 |
| 960 | 254 |
| 980 | 264 |

### Графіки залежності розв'язку від числа ітерацій

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

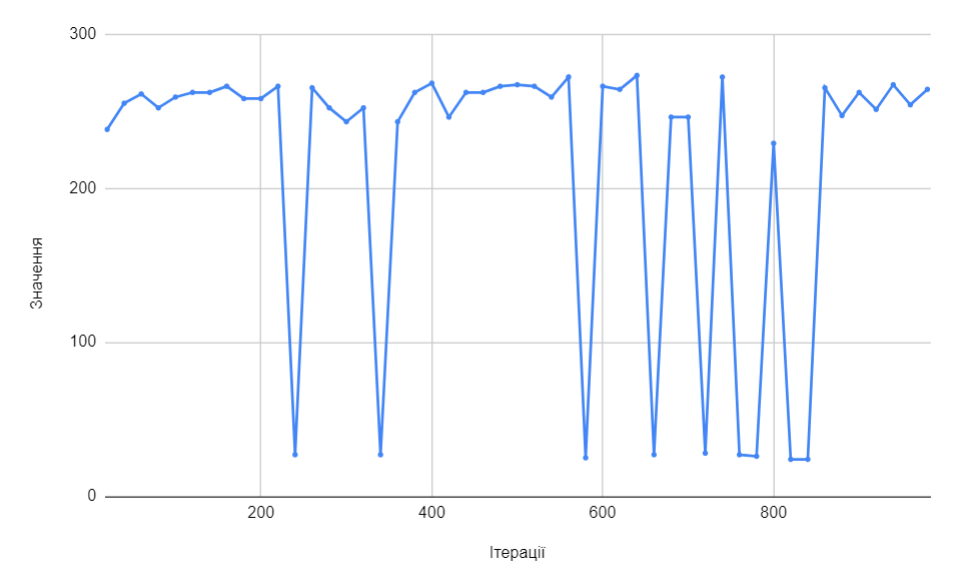


Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

Висновок

В рамках даної лабораторної роботи було вивчитено основні підходи формалізації метаеврестичних алгоритмів і вирішення типових задач з їхньою допомогою, досліджено алгоритм, вказаний у завданні. Реалізовано його на мові програмування.

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%.