**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 № 5 з дисципліни

«Проектування алгоритмів»

„**Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.2**”

**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](#_Toc52291748)

[2 Завдання 4](#_Toc52291749)

[3 Виконання 10](#_Toc52291750)

[3.1 Покроковий алгоритм 10](#_Toc52291751)

[3.2 Програмна реалізація алгоритму 10](#_Toc52291752)

[3.2.1 Вихідний код 10](#_Toc52291753)

[3.2.2 Приклади роботи 10](#_Toc52291754)

[3.3 Тестування алгоритму 11](#_Toc52291755)

[Висновок 12](#_Toc52291756)

[Критерії оцінювання 13](#_Toc52291757)

# Мета лабораторної роботи

Мета роботи – вивчити основні підходи розробки метаеврестичних алгоритмів для типових прикладних задач. Опрацювати методологію підбору прийнятних параметрів алгоритму.

# Завдання

Згідно варіанту, формалізувати алгоритм вирішення задачі відповідно загальної методології.

Записати розроблений алгоритм у покроковому вигляді. З достатнім степенем деталізації.

Виконати його програмну реалізацію на будь-якій мові програмування.

Перелік задач наведено у таблиці 2.1.

Перелік алгоритмів і досліджуваних параметрів у таблиці 2.2.

Задача і алгоритм наведені в таблиці 2.3.

Змінюючи параметри алгоритму, визначити кращі вхідні параметри алгоритму. Для цього необхідно:

* обрати критерій зупинки алгоритму (кількість ітерацій або значення ЦФ);
* зафіксувати усі параметри крім одного і змінювати цей параметр, поки не буде досягнуто пікової ефективності;
* після цього параметр фіксується і змінюються інші параметри;
* далі повторюємо процедуру спочатку, з першого зафіксованого параметру;
* зупиняємось коли будуть знайдені оптимальні параметри для даної задачі або встановлена залежність одних параметрів від інших.

Зробити узагальнений висновок в якому обов’язково описати залежність якості розв’язку від вхідних параметрів.

Таблиця 2.1 – Прикладні задачі

|  |  |
| --- | --- |
| **№** | **Задача** |
| 2 | **Задача комівояжера** (300 вершин, відстань між вершинами випадкова від 5 до 150) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів.  **Розглядається симетричний, асиметричний та змішаний варіанти.**  В загальному випадку, асиметрична задача комівояжера відрізняється тим, що ребра між вершинами можуть мати різну вагу в залежності від напряму, тобто, задача моделюється орієнтованим графом. Таким чином, окрім ваги ребер графа, слід також зважати і на те, в якому напрямку знаходяться ребра.  У випадку симетричної задачі всі пари ребер між одними й тими самими вершинами мають однакову вагу.  У випадку реальних міст може бути як симетричною, так і асиметричною в залежності від тривалості або довжини маршрутів і напряму руху.  Застосування:   * доставка товарів (в цьому випадку може бути більш доречна постановка транспортної задачі - доставка в кілька магазинів з декількох складів); * доставка води; * моніторинг об'єктів; * поповнення банкоматів готівкою; * збір співробітників для доставки вахтовим методом. |

Таблиця 2.2 – Варіанти алгоритмів і досліджувані параметри

|  |  |
| --- | --- |
| **№** | **Алгоритми і досліджувані параметри** |
| 2 | **Мурашиний алгоритм**:   * α; * β; * ρ; * Lmin; * кількість мурах М і їх типи (елітні, тощо…); * маршрути з однієї чи різних вершин. |

Таблиця 2.3 – Варіанти задач і алгоритмів

|  |  |
| --- | --- |
| **№** | **Задачі і алгоритми** |
| 7 | Задача комівояжера (симетрична мережа) + Мурашиний алгоритм |

# Виконання

## Покроковий алгоритм

1. Основний алгоритм
2. ПОЧАТОК
   1. Ініціалізувати пустий масив AllTimeBestPath
   2. Ініціалізувати число AllTimeBestDistance = Int32.MaxValue
   3. ПОКИ не виконану кількість ітерацій – вхідний параметр iterations
      1. Ініціалізація колонії мурах
      2. Розташування мурах по графу в залежності від булевого параметра \_differentPlacement
      3. Запуск обходу кожної мурахи послідовно (Пункт 4)
      4. ЦИКЛ ДЛЯ мурах у колонії
         1. ЯКЩО шлях мурахи є циклом ТА ant.\_pathDistance < AllTimeBestDistance ТО
            1. Зберегти шлях цієї мурахи у bestPath, дистанцію її шляху – у bestDistance
         2. КІНЕЦЬ ЯКЩО
      5. ЯКЩО bestDistance < AllTimeBestDistance ТО
         1. AllTimeBestPath = bestPath
         2. AllTimeBestDistance = bestDistance
      6. КІНЕЦЬ ЯКЩО
      7. Оновлення матриці феромонів
   4. КІНЕЦЬ ПОКИ
3. КІНЕЦЬ
4. Метод обходу мурахи
   1. ПОЧАТОК
      1. ПОКИ мураха має доступні для переходу вершини
         1. Отримати список доступних вершин для переходу
         2. Отримати набір ймовірностей переходу у кожну з вершин за формулою
         3. ЯКЩО мураха елітна ТО
            1. Перейти у вершину з найбільшою ймовірністю
         4. КІНЕЦЬ ЯКЩО
         5. ЯКЩО мураха звичайна ТО
            1. Отримати випадкове значення у межах [0.0, 1.0)
            2. Обрахувати циклічно вершину для переходу залежно від випадкового значення
         6. КІНЕЦЬ ЯКЩО
         7. ЯКЩО мураха дика ТО
            1. Обрати випадкову вершину без урахування ймовірностей
         8. КІНЕЦЬ ЯКЩО
      2. КІНЕЦЬ ПОКИ
   2. КІНЕЦЬ

## Програмна реалізація алгоритму

### Вихідний код

Program.cs

using System.Diagnostics;  
  
namespace Lab5;  
  
class Program  
{  
 public static void Main(string[] args)  
 {  
 Graph gr = new Graph();  
 AntColony antColony = new AntColony(g: gr, a: 1d, b: 1d, r: 0.5, lm: 500, true, elite: 0, reg: 10, wild: 0);  
 Stopwatch stopwatch = Stopwatch.StartNew();  
 antColony.Start(200);  
 Console.WriteLine($"Done in {stopwatch.Elapsed}");  
 Console.WriteLine("Best path is: ");  
 antColony.WritePath();  
 antColony.SaveResults();  
 Console.WriteLine($"All the other info is written into {AntColony.filepath}");  
 }  
}

AntColony.cs

using System.Reflection.Metadata;  
  
namespace Lab5;  
  
public class AntColony  
{  
 public Graph graph;  
 public double \_alpha;  
 public double \_beta;  
 public double \_rho;  
 public int Lmin;  
 private bool \_differentPlacement;  
 private int \_elites;  
 private int \_regs;  
 private int \_wilds;  
 private Ant[] colony;  
 private double[,] \_pheromonesMatrix;  
 private int[] AllTimeBestPath;  
 private int AllTimeBestDistance;  
 public static string filepath = "data.txt";  
  
 public AntColony(Graph g, double a, double b, double r, int lm, bool placement, int elite, int reg, int wild)  
 {  
 graph = g;  
 \_alpha = a;  
 \_beta = b;  
 \_rho = r;  
 Lmin = lm;  
 \_differentPlacement = placement;  
 \_elites = elite;  
 \_regs = reg;  
 \_wilds = wild;  
  
 \_pheromonesMatrix = new double[Graph.\_amtOfVertices, Graph.\_amtOfVertices];  
 for (int i = 0; i < Graph.\_amtOfVertices; i++)  
 {  
 for (int j = 0; j < Graph.\_amtOfVertices; j++)  
 {  
 if(graph.DistanceMatrix[i,j]>0)  
 \_pheromonesMatrix[i, j] = 0.1;  
 else  
 \_pheromonesMatrix[i, i] = 0d;  
 }  
 }  
   
 AllTimeBestDistance = Int32.MaxValue;  
 }  
  
 public void Start(int iterations)  
 {  
 int itr = 0;  
 int[] bestPath = new int[Graph.\_amtOfVertices];  
 int bestDistance;  
 while (itr < iterations)  
 {  
 itr++;  
 CreateColony();  
 PlaceAnts();  
 foreach (Ant ant in colony)  
 ant.Traverse(this);  
 bestDistance = Int32.MaxValue;  
 foreach (Ant ant in colony)  
 {  
 if (ant.IsPathValid(graph) && bestDistance > ant.\_pathDistance)  
 {  
 bestDistance = ant.\_pathDistance;  
 bestPath = ant.\_path;  
 }  
 }  
  
 if (AllTimeBestDistance > bestDistance)  
 {  
 AllTimeBestDistance = bestDistance;  
 AllTimeBestPath = bestPath;  
 }  
 PlacePheromones();  
   
 }  
 }  
  
 private void CreateColony()  
 {  
 colony = new Ant[\_elites + \_regs + \_wilds];  
 int offset = 0;  
 if (\_elites > 0)  
 {  
 for (int i = 0; i < \_elites; i++)  
 colony[i + offset] = new EliteAnt();  
 offset += \_elites;  
 }  
  
 if (\_regs > 0)  
 {  
 for (int i = 0; i < \_regs; i++)  
 colony[i + offset] = new RegularAnt();  
 offset += \_regs;  
 }  
  
 for (int i = 0; i < \_wilds; i++)  
 colony[i + offset] = new WildAnt();  
 }  
 public double[] GetChoiceProbs(int vertice, List<int> availableVertices)  
 {  
 double[] probs = new double[availableVertices.Count];  
 double sumOfProbs = 0d;   
 for (int i = 0; i < probs.Length; i++)  
 {  
 probs[i] = CalculateProbability(vertice, availableVertices[i]);  
 sumOfProbs += probs[i];  
 }  
  
 for (int i = 0; i < probs.Length; i++)  
 probs[i] /= sumOfProbs;  
  
 return probs;  
 }  
  
 private double CalculateProbability(int vertice, int probable) =>  
 Math.Pow(\_pheromonesMatrix[vertice, probable], \_alpha) \* Math.Pow(getVisibility(vertice, probable), \_beta);  
  
 double getVisibility(int i, int j) => 1d / graph.DistanceMatrix[i, j];  
   
 private void PlaceAnts()  
 {  
 Random rng = new Random();  
 if (\_differentPlacement)  
 {  
 foreach (Ant ant in colony)  
 ant.PlaceAtStart(rng.Next(0,300));  
 return;  
 }  
  
 int vertice = rng.Next(0, 300);  
 foreach (Ant ant in colony)  
 ant.PlaceAtStart(vertice);  
 }  
  
 private void PlacePheromones()  
 {  
 for (int i = 0; i < \_pheromonesMatrix.GetLength(0); i++)  
 {  
 for (int j = 0; j < \_pheromonesMatrix.GetLength(1); j++)  
 {  
 \_pheromonesMatrix[i, j] \*= (1d - \_rho);  
 }  
 }  
 foreach (Ant ant in colony)  
 {  
 for (int i = 1; i < ant.\_currentLength; i++)  
 {  
 \_pheromonesMatrix[ant.\_path[i - 1], ant.\_path[i]] += ant.pheromones[i - 1] \* ant.pheromoneCoeff;  
 }  
 }  
 }  
  
 public void WritePath()  
 {  
 for (int i = 0; i < AllTimeBestPath.Length; i++)  
 Console.Write($"{AllTimeBestPath[i]} ");  
 Console.WriteLine($"\nwith a distance of {AllTimeBestDistance}");  
 }  
  
 public void SaveResults()  
 {  
 StreamWriter writer = new StreamWriter(AntColony.filepath, false);  
 string diffplace = this.\_differentPlacement ? "yes" : "no";  
 string message = $"Alpha: {\_alpha}\nBeta: {\_beta}\nRho: {\_rho}\nL min: {Lmin}\nAnts:\n\tElite: {\_elites}\n\tRegular: {\_regs}\n\tWild: {\_wilds}\nDifferent Placement: {diffplace}";  
 writer.Write(message);  
 message = "\nPheromones matrix:\n";  
 for (int i = 0; i < \_pheromonesMatrix.GetLength(0); i++)  
 {  
 for (int j = 0; j < \_pheromonesMatrix.GetLength(1); j++)  
 message += $"{\_pheromonesMatrix[i, j]} ";  
   
 message += "\n";  
 }  
 writer.Write(message);  
 message = "\nThe best path found:\n";  
 for (int i = 0; i < AllTimeBestPath.Length; i++)  
 message += $"{AllTimeBestPath[i]} ";  
  
 message += $"\nwith a distance of {AllTimeBestDistance}";  
 writer.Write(message);  
  
 writer.Dispose();  
 }  
}

Ant.cs

using System.Net.Security;  
  
namespace Lab5;  
  
public abstract class Ant  
{  
 public int[] \_path { get; }  
 public int \_currentLength;  
 public int \_pathDistance;  
 public abstract int pheromoneCoeff { get; }  
 public double[] pheromones { get; }  
  
 public Ant()  
 {  
 \_path = new int[Graph.\_amtOfVertices];  
 pheromones = new double[Graph.\_amtOfVertices - 1];  
 \_currentLength = 0;  
 \_pathDistance = 0;  
 }  
  
 protected void SetPheromone(AntColony antColony)  
 {  
 pheromones[\_currentLength - 2] = (double)antColony.Lmin / \_pathDistance;  
 }  
  
 public void PlaceAtStart(int vertice)  
 {  
 \_path[0] = vertice;  
 \_currentLength++;  
 }  
   
 public abstract void Traverse(AntColony antColony);  
  
 protected List<int> GetAvailableVertices(AntColony antColony)  
 {  
 List<int> adjacents = antColony.graph.GetAdjacentVertices(\_path[\_currentLength - 1]);  
 for (int i = 0; i < \_currentLength; i++)  
 adjacents.Remove(\_path[i]);  
 return adjacents;  
 }  
  
 protected void MoveToVertice(AntColony antColony, List<int> adjacents, int chosenVertice)  
 {  
 \_path[\_currentLength++] = adjacents[chosenVertice];  
 \_pathDistance += antColony.graph.DistanceMatrix[\_path[\_currentLength - 2], \_path[\_currentLength - 1]];  
   
 SetPheromone(antColony);  
 }  
  
 public bool IsPathValid(Graph graph)  
 {  
 List<int> verts = graph.GetAdjacentVertices(\_path[\_currentLength - 1]);  
 return \_currentLength == Graph.\_amtOfVertices && verts.Contains(\_path[0]);  
 }  
}

EliteAnt.cs

namespace Lab5;  
  
public class EliteAnt : Ant  
{  
 public override int pheromoneCoeff => 2;  
  
 public override void Traverse(AntColony antColony)  
 {  
 while (true)   
 {  
 List<int> adjacents = GetAvailableVertices(antColony);  
 if (adjacents.Count == 0) return;  
 double[] probabilites = antColony.GetChoiceProbs(\_path[\_currentLength - 1], adjacents);  
  
 int chosenVertice = 0;  
 double maxProbability = 0d;  
  
 while (chosenVertice < probabilites.Length && maxProbability < probabilites[chosenVertice])  
 {  
 maxProbability = probabilites[chosenVertice];  
 chosenVertice++;  
 }  
  
 MoveToVertice(antColony, adjacents, chosenVertice);  
 }   
 }  
}

RegularAnt.cs

namespace Lab5;  
  
public class RegularAnt : Ant  
{  
 public override int pheromoneCoeff => 1;  
  
 public override void Traverse(AntColony antColony)  
 {  
 while (true)  
 {  
 List<int> adjacents = GetAvailableVertices(antColony);  
 if (adjacents.Count == 0) return;  
 double[] probabilites = antColony.GetChoiceProbs(\_path[\_currentLength - 1], adjacents);  
 Random rng = new Random();  
 double randomChoice = rng.NextDouble();  
 int chosenVertice = 0;  
 if (probabilites.Length > 1)  
 {  
 for (int i = 1; i < probabilites.Length; i++)  
 probabilites[i] += probabilites[i - 1];  
   
 while (chosenVertice < probabilites.Length && randomChoice > probabilites[chosenVertice])  
 chosenVertice++;  
 }  
  
 MoveToVertice(antColony, adjacents, chosenVertice);  
 }   
 }  
}

WildAnt.cs

namespace Lab5;  
  
public class WildAnt : Ant  
{  
 public override int pheromoneCoeff => 1;  
  
 public override void Traverse(AntColony antColony)  
 {  
 while (true)  
 {  
 List<int> adjacents = GetAvailableVertices(antColony);  
 if (adjacents.Count == 0) break;  
 Random rng = new Random();  
 int chosenVertice = rng.Next(0, adjacents.Count);  
  
 MoveToVertice(antColony, adjacents, chosenVertice);  
 }  
 }  
}

Graph.cs

namespace Lab5;  
  
public class Graph  
{  
 private static string path = "graph.txt";  
 public static int \_amtOfVertices = 300;  
 public int[,] DistanceMatrix { get; }  
  
 public Graph()  
 {  
 DistanceMatrix = new int[\_amtOfVertices, \_amtOfVertices];  
 for (int k = 0; k < \_amtOfVertices; k++)  
 {  
 for (int l = 0; l < \_amtOfVertices; l++)  
 {  
 DistanceMatrix[k, l] = -1;  
 }  
 DistanceMatrix[k, k] = 0;  
 }  
  
 int weight = 0;  
 StreamReader reader = new StreamReader(path);  
 for (int i = 0; i < \_amtOfVertices; i++)  
 {  
 string[] line = reader.ReadLine().Split();  
 for (int j = 0; j < \_amtOfVertices; j++)  
 {  
 DistanceMatrix[i,j] = Int32.Parse(line[j]);  
 }  
 }  
 reader.Dispose();  
 }  
  
 public List<int> GetAdjacentVertices(int vertice)  
 {  
 List<int> adjacentVertices = new List<int>();  
 for (int i = 0; i < \_amtOfVertices; i++)  
 if (vertice != i && DistanceMatrix[vertice, i] != -1) adjacentVertices.Add(i);  
  
 return adjacentVertices;  
 }  
   
}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

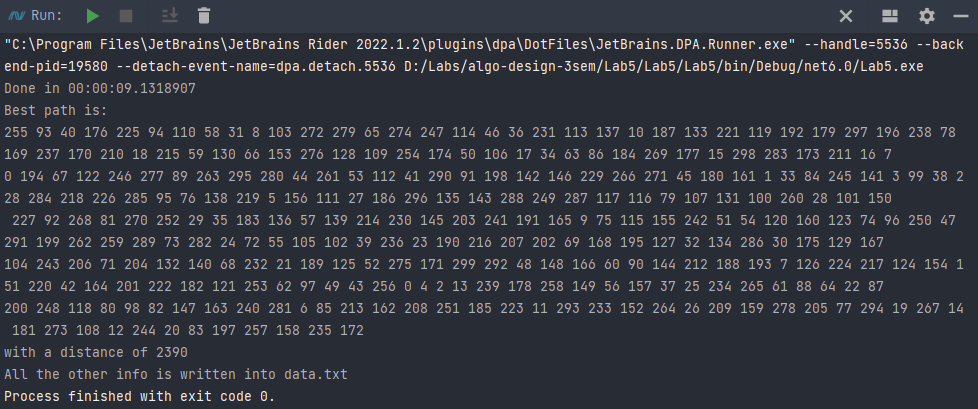


Рисунок 3.1 – Виведення найкращого шляху

Text

Description automatically generated

Рисунок 3.2 – Збереження додаткової інформації у файл data.txt

## Тестування алгоритму

Перед початком тестування необхідно означити: весь код працює на основі одного й того ж графа, збереженого у текстовому файлі graph.txt, що підпадає під умову Дірака: степінь кожної вершини є не меншою за загальну кількість вершин. До того ж опишемо деякі сталі значення, які будуть незмінними протягом усього процесу тестування:

1. Кількість ітерацій - 200
2. Кількість вершин – 300
3. Нижня межа відстані між вершинами – 5
4. Верхня межа відстані між вершинами – 150

Далі зафіскуємо початкові значення параметрів з таблиці 2.2, з яких почнеться наше дослідженння. Дані початкові значення наведено у таблиці 3.3

Таблиця 3.3 – Початкові значення досліджуваних параметрів

|  |  |
| --- | --- |
| Параметр | Значення |
| α | 1.0 |
| β | 1.0 |
| ρ | 0.5 |
| Lmin | 500 |
| Мелітні | 0 |
| Мзвичайні | 10 |
| Мдикі | 0 |
| Маршрути з однієї чи різних вершин | З різних вершин |

### Дослідження параметру α

Параметр α відповідає за вагу значення феромона при обрахуванні ймовірності переходу в певну вершину. Набором значень α будуть числа від 0.0 до 3.0 з кроком 0.5. Інші параметри будуть узяті з таблиці 3.3. Для кожного значення α проведемо 5 дослідів і визначимо середній результат для поточного значення.

Таблиця 3.3 – Значення цільової функції при різних значеннях α

|  |  |  |
| --- | --- | --- |
| Значення α | Номер дослідження | Значення цільової функції |
| 0 | 1 | 11024 |
| 2 | 10367 |
| 3 | 10299 |
| 4 | 10706 |
| 5 | 10707 |
| **Середнє** | 10621 |
| 0.5 | 1 | 2993 |
| 2 | 3043 |
| 3 | 2988 |
| 4 | 2958 |
| 5 | 3120 |
| **Середнє** | 3020 |
| 1 | 1 | 2308 |
| 2 | 2336 |
| 3 | 2309 |
| 4 | 2373 |
| 5 | 2449 |
| **Середнє** | 2355 |
| 1.5 | 1 | 3157 |
| 2 | 3290 |
| 3 | 3332 |
| 4 | 3133 |
| 5 | 3537 |
| **Середнє** | 3290 |
| 2 | 1 | 4218 |
| 2 | 4558 |
| 3 | 4470 |
| 4 | 3957 |
| 5 | 3878 |
| **Середнє** | 4216 |
| 2.5 | 1 | 6061 |
| 2 | 6168 |
| 3 | 6012 |
| 4 | 5882 |
| 5 | 6203 |
| **Середнє** | 6065 |
| 3 | 1 | 6416 |
| 2 | 6418 |
| 3 | 6640 |
| 4 | 6949 |
| 5 | 6105 |
| **Середнє** | 6506 |

Діаграма 3.1 – Значення цільової функції при різних значеннях α

З діаграми розуміємо, що оптимальним значенням α є 1. При α=0 алгоритм стає жадібним, що означає що мурахи найімовірніше перейдуть до найближчої вершини, такий варіант не є надто точним для нашої задачі. Занадто великі значення теж є неоптимальними.

### Дослідження параметру β

Параметр β відповідає за вагу значення видимості вершини (значення, обернене відстані до цієї вершини) при обрахуванні ймовірності переходу в певну вершину. Набором значень β будуть числа від 0.0 до 3.0 з кроком 0.5. Інші параметри будуть узяті з таблиці 3.3. Для кожного значення β проведемо 5 дослідів і визначимо середній результат для поточного значення.

Таблиця 3.4 – Значення цільової функції при різних значеннях β

|  |  |  |
| --- | --- | --- |
| Значення β | Номер дослідження | Значення цільової функції |
| 0 | 1 | 18386 |
| 2 | 18470 |
| 3 | 18753 |
| 4 | 18232 |
| 5 | 18511 |
| **Середнє** | 18470 |
| 0.5 | 1 | 3132 |
| 2 | 3340 |
| 3 | 3319 |
| 4 | 3367 |
| 5 | 3612 |
| **Середнє** | 3354 |
| 1 | 1 | 2345 |
| 2 | 2354 |
| 3 | 2358 |
| 4 | 2328 |
| 5 | 2209 |
| **Середнє** | 2319 |
| 1.5 | 1 | 2118 |
| 2 | 2108 |
| 3 | 2096 |
| 4 | 2217 |
| 5 | 2073 |
| **Середнє** | 2142 |
| 2 | 1 | 2084 |
| 2 | 2090 |
| 3 | 2022 |
| 4 | 2077 |
| 5 | 2129 |
| **Середнє** | 2080 |
| 2.5 | 1 | 2129 |
| 2 | 2003 |
| 3 | 2105 |
| 4 | 2065 |
| 5 | 2103 |
| **Середнє** | 2081 |
| 3 | 1 | 2030 |
| 2 | 2143 |
| 3 | 2089 |
| 4 | 2059 |
| 5 | 2125 |
| **Середнє** | 2089 |

Діаграма 3.2 – Значення цільової функції при різних значеннях β

Бачимо, що різке підвищення ефективності спостерігається при переході зі значень β=0 і β=0.5. При β=0 мурахи починають орієнтуватись суто на «запах» феромону, що теж виявилось вкрай неефективним, оскільки на початку всі дуги мають однакову кількість феромону. Далі зміна значення бета не грає важливої ролі. І все ж, з мінімальним розривом найоптимальнішим виявилось значення 2.5.

### Дослідження набору параметрів α і β

В попередніх дослідах ми розглядали значення α і β по окремості, беручи початкові значення інших параметрів. Тепер поглянемо на взаємодію цих двох величин: набором значень є декартовий добуток наборів α від 0 до 3 через 0.5 і β від 0 до 3 через 0.5. Усі інші значення беруться з таблиці початкових значень.

Таблиця 3.5 – Значення цільової функції при різних значеннях α та β

|  |  |  |
| --- | --- | --- |
| Значення α | Значення β | Значення цільової функції |
| 0 | 0 | 21142 |
| 0 | 0.5 | 16080 |
| 0 | 1 | 10737 |
| 0 | 1.5 | 6987 |
| 0 | 2 | 4663 |
| 0 | 2.5 | 3566 |
| 0 | 3 | 3041 |
| 0.5 | 0 | 20461 |
| 0.5 | 0.5 | 5221 |
| 0.5 | 1 | 3037 |
| 0.5 | 1.5 | 2573 |
| 0.5 | 2 | 2490 |
| 0.5 | 2.5 | 2253 |
| 0.5 | 3 | 2164 |
| 1 | 0 | 18498 |
| 1 | 0.5 | 3311 |
| 1 | 1 | 2364 |
| 1 | 1.5 | 2158 |
| 1 | 2 | 2167 |
| 1 | 2.5 | 2006 |
| 1 | 3 | **2004** |
| 1.5 | 0 | 21086 |
| 1.5 | 0.5 | 6796 |
| 1.5 | 1 | 3479 |
| 1.5 | 1.5 | 2582 |
| 1.5 | 2 | 2402 |
| 1.5 | 2.5 | 2178 |
| 1.5 | 3 | 2151 |
| 2 | 0 | 21250 |
| 2 | 0.5 | 9984 |
| 2 | 1 | 4216 |
| 2 | 1.5 | 3092 |
| 2 | 2 | 2533 |
| 2 | 2.5 | 2328 |
| 2 | 3 | 2216 |
| 2.5 | 0 | 21524 |
| 2.5 | 0.5 | 12760 |
| 2.5 | 1 | 5940 |
| 2.5 | 1.5 | 3163 |
| 2.5 | 2 | 2775 |
| 2.5 | 2.5 | 2493 |
| 2.5 | 3 | 2486 |
| 3 | 0 | 22360 |
| 3 | 0.5 | 12674 |
| 3 | 1 | 7664 |
| 3 | 1.5 | 3567 |
| 3 | 2 | 3099 |
| 3 | 2.5 | 2541 |
| 3 | 3 | 2443 |

Таким чином, взаємно оптимальними значеннями для α і β є 1 і 3 відповідно. Для наступних досліджень використовуватимемо саме ці значення. Також спостерігається, що при α=0 або β=0 спостерігаються погані результати, в цілому вони покращуються, коли α>0 і β>0.

### Дослідження параметру ρ

Наступний параметр у серії досліджень – ρ. Він визначає швидкість випаровування феромонів на усьому графі: з кожною ітерацією старі значення феромонів зменшуються у разів. Проведемо серію вимірювань при α=1, β=3, інші значення – з таблиці 3.3, ρ – від 0 до 1 з кроком 0.2.

Таблиця 3.6 – Значення цільової функції при різних значеннях ρ

|  |  |  |
| --- | --- | --- |
| Значення ρ | Номер дослідження | Значення цільової функції |
| 0 | 1 | 2071 |
| 2 | 2084 |
| 3 | 2095 |
| 4 | 2107 |
| 5 | 2093 |
| **Середнє** | 2090 |
| 0.2 | 1 | 2095 |
| 2 | 1992 |
| 3 | 2144 |
| 4 | 2103 |
| 5 | 2069 |
| **Середнє** | 2081 |
| 0.4 | 1 | 2000 |
| 2 | 2079 |
| 3 | 2157 |
| 4 | 2072 |
| 5 | 2108 |
| **Середнє** | 2083 |
| 0.6 | 1 | 2143 |
| 2 | 1981 |
| 3 | 2100 |
| 4 | 2083 |
| 5 | 2101 |
| **Середнє** | 2082 |
| 0.8 | 1 | 1985 |
| 2 | 2132 |
| 3 | 2074 |
| 4 | 2147 |
| 5 | 2104 |
| **Середнє** | 2088 |
| 1 | 1 | 2852 |
| 2 | 2809 |
| 3 | 2974 |
| 4 | 2962 |
| 5 | 2809 |
| **Середнє** | 2881 |

Діаграма 3.3 – Значення цільової функції при різних значеннях ρ

Отже, чітко видно, що найважливішим критерієм графіку є ρ<1, але строго математично маємо найоптимальніше значення ρ=0.2. При моментальному випаровуванні (ρ=1) мурахи не відчувають феромону, залишеного попередніми популяціями, тому результати – найгірші. Надалі використовуватимемо оптимальне значення ρ.

### Дослідження параметру Lmin

Наступним параметром у задачі є Lmin – ціна передбачуваного ідеального рішення проблеми. Параметр фігурує у формулі кількості феромону, що відкладає кожна мураха (). Тому чим більше очікуване ідеальне рішення, тим більше відкладатиме феромону мураха. Проводитимемо дослідження при знайдених раніше оптимальних рішеннях, інші – з таблиці 3.3. Діапазон Lmin – від 500 до 5000 з кроком 500.

Таблиця 3.7 – Значення цільової функції при різних значеннях Lmin

|  |  |  |
| --- | --- | --- |
| Значення Lmin | Номер дослідження | Значення цільової функції |
| 500 | 1 | 2113 |
| 2 | 2084 |
| 3 | 2083 |
| 4 | 2008 |
| 5 | 2128 |
| **Середнє** | 2083 |
| 1000 | 1 | 2122 |
| 2 | 2103 |
| 3 | 2058 |
| 4 | 2127 |
| 5 | 2119 |
| **Середнє** | 2106 |
| 1500 | 1 | 2109 |
| 2 | 2061 |
| 3 | 2146 |
| 4 | 2144 |
| 5 | 2099 |
| **Середнє** | 2112 |
| 2000 | 1 | 2115 |
| 2 | 2052 |
| 3 | 2029 |
| 4 | 2102 |
| 5 | 2092 |
| **Середнє** | 2078 |
| 2500 | 1 | 2101 |
| 2 | 2107 |
| 3 | 2052 |
| 4 | 2077 |
| 5 | 2094 |
| **Середнє** | 2086 |
| 3000 | 1 | 2103 |
| 2 | 2063 |
| 3 | 2124 |
| 4 | 2069 |
| 5 | 2088 |
| **Середнє** | 2089 |
| 3500 | 1 | 2061 |
| 2 | 2101 |
| 3 | 2089 |
| 4 | 2063 |
| 5 | 2094 |
| **Середнє** | 2082 |
| 4000 | 1 | 2043 |
| 2 | 2105 |
| 3 | 2117 |
| 4 | 2014 |
| 5 | 2142 |
| **Середнє** | 2084 |
| 4500 | 1 | 2101 |
| 2 | 2140 |
| 3 | 2059 |
| 4 | 2103 |
| 5 | 2065 |
| **Середнє** | 2094 |
| 5000 | 1 | 2113 |
| 2 | 2086 |
| 3 | 2098 |
| 4 | 2137 |
| 5 | 2152 |
| **Середнє** | 2117 |

Діаграма 3.4 – Значення цільової функції при різних значеннях Lmin

Цей параметр не дає чітких тенденцій, видно, що параметр не сильно впливає на ефективність алгоритму, але зрозуміло, що найоптимальнішим є значення Lmin=2000. Записуємо його до інших оптимальних рішень та використовуємо далі.

### Дослідження методу розташування мурaх

Наступний параметр не є скалярним, проте не менш важливий за попередні. Розташування мурах можливе у одній або різних вершинах. Проведемо по 10 дослідів на обох варіантах, використовуємо знайдені раніше оптимальні параметри, інші – з таблиці 3.3.

Таблиця 3.8 – Значення цільової функції при різних розміщеннях мурах

|  |  |  |
| --- | --- | --- |
| Розташування мурах | Номер дослідження | Значення цільової функції |
| На різних вершинах | 1 | 2027 |
| 2 | 2072 |
| 3 | 2064 |
| 4 | 2117 |
| 5 | 2084 |
| 6 | 2108 |
| 7 | 2079 |
| 8 | 2019 |
| 9 | 2114 |
| 10 | 2060 |
| **Середнє** | 2074.4 |
| На одній вершині | 1 | 2065 |
| 2 | 2093 |
| 3 | 2117 |
| 4 | 2073 |
| 5 | 2051 |
| 6 | 2029 |
| 7 | 2114 |
| 8 | 2105 |
| 9 | 2100 |
| 10 | 2155 |
| **Середнє** | 2090.2 |

Діаграма 3.5 – Значення цільової функції при різних розміщеннях мурах

Бачимо прекрасні результати роботи при обох варіантах, проте розташування на різних вершинах все ж перемагає. Вважаємо таке розміщення оптимальним.

### Дослідження параметрів М, Мелітні, Мзвичайні, Мдикі

Останніми параметрами є загальна кількість мурах М, та окрема кількість кожного з видів. Спочатку дослідимо оптимальну кількість мурах в загальному, використовуючи вид звичайних мурах. Діапазон візьмемо від 1 до 200, крок експоненційний.

Таблиця 3.9 – Значення цільової функції при різній кількості мурах М

|  |  |  |
| --- | --- | --- |
| Значення М | Номер дослідження | Значення цільової функції |
| 1 | 1 | 2379 |
| 2 | 2227 |
| 3 | 2244 |
| 4 | 2327 |
| 5 | 2283 |
| **Середнє** | 2292 |
| 2 | 1 | 2182 |
| 2 | 2197 |
| 3 | 2291 |
| 4 | 2173 |
| 5 | 2208 |
| **Середнє** | 2210.2 |
| 5 | 1 | 2106 |
| 2 | 2124 |
| 3 | 2096 |
| 4 | 2133 |
| 5 | 2097 |
| **Середнє** | 2111.2 |
| 10 | 1 | 2067 |
| 2 | 2048 |
| 3 | 2111 |
| 4 | 2112 |
| 5 | 2105 |
| **Середнє** | 2088.6 |
| 20 | 1 | 2067 |
| 2 | 2050 |
| 3 | 2004 |
| 4 | 2068 |
| 5 | 2077 |
| **Середнє** | 2053.2 |
| 50 | 1 | 2073 |
| 2 | 1994 |
| 3 | 2052 |
| 4 | 2047 |
| 5 | 2053 |
| **Середнє** | 2043.8 |
| 100 | 1 | 2000 |
| 2 | 1971 |
| 3 | 2009 |
| 4 | 2000 |
| 5 | 1986 |
| **Середнє** | 1993.2 |
| 200 | 1 | 1990 |
| 2 | 1941 |
| 3 | 2004 |
| 4 | 1972 |
| 5 | 2004 |
| **Середнє** | 1982.2 |

Діаграма 3.6 – Значення цільової функції при різній кількості мурах М

Загальна тенденція цілком зрозуміла та логічна – чим більше мурах в колонії – тим краще результат, тому вносимо кількість мурах М=200 до оптимальних значень.

Тепер дослідимо кількість елітних мурах Мелітні. Діапазон – від 0 до 200, інші мурахи – звичайні.

Таблиця 3.10 – Значення цільової функції при різних значеннях Мелітні

|  |  |  |
| --- | --- | --- |
| Значення Мзвичайні | Значення Мелітні | Значення цільової функції |
| 200 | 0 | 1994 |
| 199 | 1 | 1984 |
| 198 | 2 | 1989 |
| 195 | 5 | 2013 |
| 190 | 10 | 2012 |
| 180 | 20 | 2056 |
| 150 | 50 | 2035 |
| 100 | 100 | 2080 |
| 0 | 200 | 2097 |

Отже, найкращі результати виявились при наборі зі 199 звичайних та 1 елітної мурахи. Таку кількість елітних мурах використовуватимемо і надалі.

Тепер протестуємо кількість диких мурах Мдикі. Розглядуватимемо 2 випадки, коли є 1 елітна мураха і коли їх нема зовсім. У першому випадку діапазон кількості диких мурах буде від 0 до 199, а кількість звичайних мурах варіюватиметься від 200 до 0. Другий випадок – окрема єдина ситуація, коли маємо колонію з 200 диких мурах.

Таблиця 3.11 – Значення цільової функції при різних значеннях Мдикі

|  |  |  |  |
| --- | --- | --- | --- |
| Мзвичайні | Мелітні | Мдикі | Значення цільової функції |
| 199 | 1 | 0 | 2003 |
| 198 | 1 | 1 | 2015 |
| 197 | 1 | 2 | 2031 |
| 194 | 1 | 5 | 1972 |
| 189 | 1 | 10 | 1983 |
| 179 | 1 | 20 | 1968 |
| 149 | 1 | 50 | 2031 |
| 99 | 1 | 100 | 2036 |
| 0 | 1 | 199 | 2333 |
| 0 | 0 | 200 | 20090 |

Ось таким чином, через серію дослідів, отримали оптимальні значення усіх досліджуваних параметрів задачі комівояжера. Їх значення наведені нижче у таблиці 3.12.

Таблиця 3.12 – Оптимальні значення досліджуваних параметрів

|  |  |
| --- | --- |
| Параметр | Значення |
| α | 1.0 |
| β | 3.0 |
| ρ | 0.2 |
| Lmin | 2000 |
| Мелітні | 1 |
| Мзвичайні | 179 |
| Мдикі | 20 |
| Маршрути з однієї чи різних вершин | З різних вершин |

Висновок

В рамках даної лабораторної роботи дослідив ціль задачі комівояжера, реалізував пошук рішення за допомогою алгоритму мурашиної колонії (ACO), а саме:

* Розглянув концепцію мурашиної колонії
* Побудував граф що підходить для задачі
* Реалізував поведінку різних видів мурах
* Реалізував основний алгоритм
* Провів серію досліджень з пошуку оптимальних значень параметрів мурашиного алгоритму.

Критерії оцінювання

При здачі лабораторної роботи до 11.12.2022 включно максимальний бал дорівнює – 5. Після 11.12.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* покроковий алгоритм – 15%;
* програмна реалізація алгоритму – 50%;
* тестування алгоритму– 30%;
* висновок – 5%.