**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 № 3 з дисципліни

«Проектування алгоритмів»

„**Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-01 Черпак Андрій Вадимович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2021

Зміст

[1 Мета лабораторної роботи 3](#_Toc51260917)

[2 Завдання 4](#_Toc51260918)

[3 Виконання 9](#_Toc51260919)

[3.1 Програмна реалізація алгоритму 9](#_Toc51260920)

[3.1.1 Вихідний код 9](#_Toc51260921)

[3.1.2 Приклади роботи 9](#_Toc51260922)

[3.2 Тестування алгоритму 10](#_Toc51260923)

[3.2.1 Значення цільової функції зі збільшенням кількості ітерацій 10](#_Toc51260924)

[3.2.2 Графіки залежності розв'язку від числа ітерацій 10](#_Toc51260925)

[Висновок 11](#_Toc51260926)

[Критерії оцінювання 12](#_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** |
| 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). |

# Виконання

## Програмна реалізація алгоритму

### Вихідний код

public void ACO(int numberOfIterations)  
{  
 for (int ictr = 0; ictr < numberOfIterations; ictr++)  
 {  
 List<Ant> ants = new();  
 while (ants.Count<M)  
 {  
 int p = rand.Next(D.Length);  
 ants.Add(new Ant(D, T, p));  
 }  
 updateT(ants);  
if ((ictr + 1) % 20 == 0)  
 {  
Console.WriteLine("\r"+ictr+" : "+bestL);  
}  
 }

getResult();

}

public void getResult()  
{  
 string path = $"[{bestPath[0]}";  
 for(int i = 1; i<bestPath.Count; i++)  
 {  
 path += $" ==> {bestPath[i]}";  
 }  
 path += $"], {bestL}";  
 Console.WriteLine("\r"+path);  
}

private void updateT(List<Ant> ants)  
{  
 for (int i = 0; i<T.Length; i++) for (int j = 0; j < T[i].Length; j++) T[i][j] \*= 1 - p;  
 foreach (Ant ant in ants)  
 {  
 if (ant.Lk < bestL)  
 {  
 bestL = ant.Lk;  
 bestPath = ant.path;  
 }  
 for (int i = 1; i < ant.path.Count; i++)  
 {  
 T[ant.path[i - 1]][ant.path[i]] += (double)Lmin / ant.Lk;  
 T[ant.path[i]][ant.path[i-1]] += (double)Lmin / ant.Lk;  
 }  
 }  
}

public Ant(int[][] D, double[][] T, int startPoint)  
{  
 Lk = 0;  
 int current = startPoint;  
 path = new();  
 path.Add(current);  
 visited = new int[D.Length];  
 visited[current] = 1;  
 while (path.Count < D.Length)  
 {  
 int next = chooseNext(D[current], T[current], visited);  
 path.Add(next);  
 Lk += D[current][next];  
 visited[next] = 1;  
 current = next;  
 }  
 Lk += D[current][startPoint];  
 path.Add(startPoint);  
}

private static int chooseNext(int[] D, double[] T, int[] visited)  
{  
 List<KeyValuePair<int, double>> \_possible = new List<KeyValuePair<int, double>>();  
 double denominator = 0;  
 for (int i = 0; i < D.Length; i++)  
 {  
 if (visited[i] == 0)  
 {  
 double numerator = Math.Pow(T[i], a) \* Math.Pow(1.0 / D[i], b);  
 if (numerator < Math.Pow(10, -15)) numerator = Math.Pow(10, -15);  
 \_possible.Add(new KeyValuePair<int, double>(i, numerator));  
 denominator += numerator;  
 }  
 }  
 double test = 0;  
 List<KeyValuePair<int, double>> possible = new List<KeyValuePair<int, double>>();  
 for (int i = 0; i < \_possible.Count; i++)  
 {  
 test += \_possible[i].Value / denominator;  
 possible.Add(new KeyValuePair<int, double>(\_possible[i].Key, \_possible[i].Value/denominator));  
 }  
 if (Math.Round(test, 8)!=1) Console.WriteLine($"{test}!=1");  
 double choise = rand.NextDouble();  
 foreach (var vertice in possible)  
 {  
 if (choise < vertice.Value) return vertice.Key;  
 choise -= vertice.Value;  
 }  
 return -1;  
}

public static void Main()  
{  
MatrixFactory mf = new MatrixFactory(@"file2.csv");  
 int[][] D = mf.getD();  
 double[][] T = mf.getT();  
 ACO aco = new(D, T);  
 Console.WriteLine($"Lmin = {aco.Lmin}");  
 aco.Go(1000);  
}

public int[][] getD()  
{  
 using (StreamReader sr = new StreamReader(filename, System.Text.Encoding.Default))  
 {  
 graphSize = Int32.Parse(sr.ReadLine());  
 int[][] D = new int[graphSize][];  
 for (int i = 0; i < graphSize; i++)  
 {  
 D[i] = new int[graphSize];  
 String[] strDistances = sr.ReadLine().Split(new[] {','}, StringSplitOptions.**RemoveEmptyEntries**);  
 for (int j = 0; j < graphSize; j++) D[i][j] = Int32.Parse(strDistances[j]);  
 }  
 return D;  
 }  
}  
  
public double[][] getT()  
{  
 double[][] T = new double[graphSize][];  
 Random rand = new();  
 for (int i = 0; i < graphSize; i++)  
 {  
 T[i] = new double[graphSize];  
 for (int j = 0; j < graphSize; j++)  
 {  
 if (i == j) T[i][j] = 0;  
 else T[i][j] = 3;  
 }  
 }  
  
 return T;  
}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

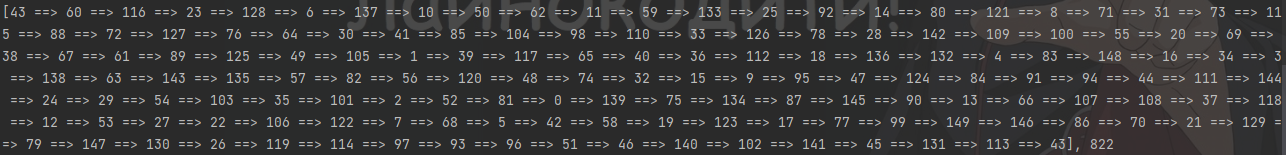


Рисунок 3.1 – результат роботи алгоритму

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

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

|  |  |
| --- | --- |
| Ітерація | Значення цільової функції |
| 19 | 881 |
| 39 | 839 |
| 59 | 833 |
| 79 | 825 |
| 99 | 825 |
| 119 | 825 |
| 139 | 825 |
| 159 | 825 |
| 179 | 825 |
| 199 | 825 |
| 219 | 822 |
| 239 | 822 |
| 259 | 822 |
| 279 | 822 |
| 299 | 822 |
| 319 | 822 |
| 339 | 822 |
| 359 | 822 |
| 379 | 822 |
| 399 | 822 |
| 419 | 822 |
| 439 | 822 |
| 459 | 822 |
| 479 | 822 |
| 499 | 822 |
| 519 | 822 |
| 539 | 822 |
| 559 | 822 |
| 579 | 822 |
| 599 | 822 |
| 619 | 822 |
| 639 | 822 |
| 659 | 822 |
| 679 | 822 |
| 699 | 822 |
| 719 | 822 |
| 739 | 822 |
| 759 | 822 |
| Ітерація | Значення цільової функції |
| 779 | 822 |
| 799 | 822 |
| 819 | 822 |
| 839 | 822 |
| 859 | 822 |
| 879 | 822 |
| 899 | 822 |
| 919 | 822 |
| 939 | 822 |
| 959 | 822 |
| 979 | 822 |
| 999 | 822 |

Таблиця 3.2 – Значення цільової функції залежно від кількості ітерацій

### 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 5.11.2021 включно максимальний бал дорівнює – 5. Після 5.11.2021 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* програмна реалізація алгоритму – 75%;
* тестування алгоритму– 20%;
* висновок – 5%.