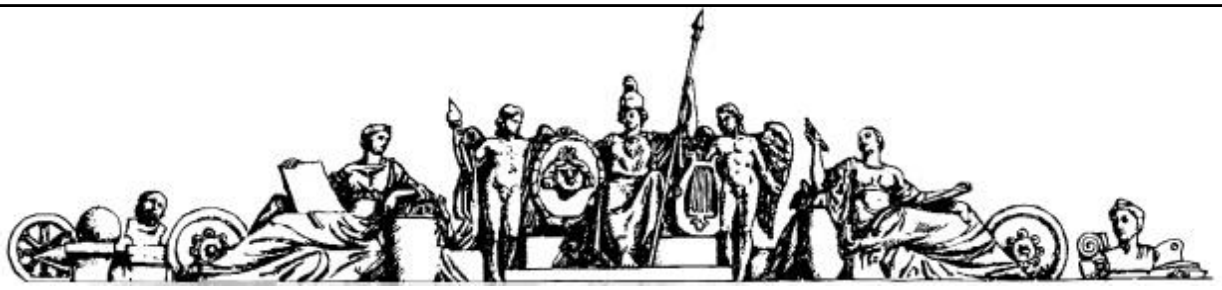

Государственное образовательное учреждение высшего профессионального образования
«Московский Государственный Технический Университет имени Н.Э. Баумана»

ОТЧЕТ

По лабораторной №7

По курсу «Анализ алгоритмов»

на тему «Муравьиный алгоритм»

Исполнитель	Студент:	Богунов Б.М.
	Группа:	ИУ7-54
Принял	Преподаватель:	Волкова Л. Л.

Москва 2017

Оглавление

Постановка задачи

Произвести реализацию муравьиного алгоритма для задачи коммивояжёра и сделать выводы о проделанной работе.

1. Ввод матрицы расстояний D
2. Инициализация параметров алгоритма – $Qe,,,\beta\alpha$
3. Инициализация рёбер – присвоение видимости $ij\eta$ и начальной концентрации феромона
4. Размещение муравьёв в случайно выбранные города без совпадений
5. Выбор начального кратчайшего маршрута и определение L^*

//Основной цикл

6. Цикл по времени жизни колонии $t=1$, $\max t$
7. Цикл по всем муравьям $k=1,m$
8. Построить маршрут по правилу (1) и рассчитать длину $)(tTk)(tLk$
9. конец цикла по муравьям
10. Проверка всех на лучшее решение по сравнению с $L^*)(tLk$
11. В случае если решение лучше, обновить L^* и $T^*)(tLk$
12. Цикл по всем рёбрам графа
13. Обновить следы феромона на ребре по правилам (2) и (3)
14. конец цикла по рёбрам
15. конец цикла по времени
16. Вывести кратчайший маршрут T^* и его длину L^*

Листинг кода

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab7
{
    public class Ant
    {
        public int curPos;
        public List<int> toVisit = new List<int>();
        public List<int> visited = new List<int>();
        public double distance = 0;

        public Ant(int cur_pos, int num)
        {
            curPos = cur_pos;
            for (int i = 0; i < num; i++)
            {
                if (i != curPos)
                    toVisit.Add(i);
            }
            visited.Add(curPos);
        }
    }

    class Program
    {
        private static double[,] GenGraph()
        {
            Random r = new Random();
            int size = r.Next(3, 10);
            double[,] D = new double[size, size];

            for (int i = 0; i < size; i++)
            {
                for (int j = i; j < size; j++)
                {
                    if (i == j)
                        D[i, j] = 0;
                    else
                        D[i, j] = r.Next(1, 50) * r.NextDouble();
                }
            }

            for (int j = 1; j < size; j++)
            {
                for (int i = 0; i < j; i++)
                {
                    D[j, i] = D[i, j];
                }
```

```
    }

    return D;
}

private static double[,] CalcVision(double [,] D)
{
    int size = D.GetLength(0);
    double[,] eta = new double[size, size];

    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++)
        {
            if (i == j)
                eta[i, j] = 1;
            else
                eta[i, j] = 1 / D[i, j];
        }

    return eta;
}

private static double[,] SetBegFeromon(double[,] D)
{
    int size = D.GetLength(0);
    double[,] teta = new double[size, size];

    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++)
        {
            if (i == j)
                teta[i, j] = 0;
            else
                teta[i, j] = 0.1;
        }

    return teta;
}

private static double CalcDistance(List<int> T, double[,] D)
{
    double distance = 0;

    T.Add(T[0]);
    for (int i = 1; i < T.Count; i++)
    {
        distance += D[T[i - 1], T[i]];
    }
    T.RemoveAt(T.Count-1);

    return distance;
}

static void Main(string[] args)
{
    Random r = new Random();

    double[,] D = new double[,] { { 0, 3, 5.3, 7.1, 7.6, 7.1,5},
```

```
                                    { 3, 0, 2.7, 5.3, 7,7.2,5.8 },
                                    { 5.3, 2.7, 0, 3, 5.3,6.4,5.8 },
                                    { 7.1, 5.3, 3, 0, 2.7,4.4,5 },
                                    { 7.6, 7, 5.3, 2.7, 0,2,3.6 },
                                    { 7.1, 7.2, 6.4, 4.4, 2,0,2.3 },
                                    { 5, 5.8, 5.8, 5, 3.6,2.3,0 } };
            double alpha = 2, beta = 1, Q = 10; // Параметры алгоритма
            var eta = CalcVision(D); // Присвоение видимости
            var teta = SetBegFeromon(D); // Задать начальное значение ферамонов
            double p = 0.5; // Коеффициент испарения
            int timeMax = 10; // Максимальное время
            int size = D.GetLength(0);

            List<int> T = new List<int>(); // Текущий лучший путь

            List<int> Temp = new List<int>();
            Random randid = new Random();
            for (int i = 0; i < D.GetLength(0); i++)
            {
                Temp.Add(i);
            }
            for (int i = 0; i < D.GetLength(0); i++)
            {
                int id = randid.Next(0, Temp.Count);
                T.Add(Temp[id]);
                Temp.RemoveAt(id);
            }
            double L = CalcDistance(T, D); // Текущая длина пути

            List<Ant> antsList = new List<Ant>();
            for (int time = 0; time < timeMax; time++)
            {
                for (int i = 0; i < size; i++)
                    antsList.Add(new Ant(i, size));

                double[,] dteta = new double[size, size]; // Приращение ферамонов на
каждом ребре
                for (int k = 0; k < antsList.Count; k++) // По всем муравьям
                {
                    while (antsList[k].toVisit.Count != 0)
                    {
                        //Вычисление P(вероятность того, куда он скорее всего пойдет)
                        int numToVis = antsList[k].toVisit.Count;
                        double[] P = new double[numToVis];
                        double sum = 0;

                        for (int l = 0; l < numToVis; l++)
                        {
                            sum += Math.Pow(teta[antsList[k].curPos,
antsList[k].toVisit[l]], alpha) * Math.Pow(eta[antsList[k].curPos,
antsList[k].toVisit[l]], beta);
                        }

                        for (int j = 0; j < numToVis; j++)
                        {
                            P[j] = Math.Pow(teta[antsList[k].curPos,
antsList[k].toVisit[j]], alpha) * Math.Pow(eta[antsList[k].curPos,
antsList[k].toVisit[j]], beta) / sum;
```

```
                    }

                    double t = r.NextDouble();
                    double chanceLine = 0;
                    for (int i = 0; i < numToVis; i++)
                    {
                        chanceLine += P[i];
                        if (chanceLine >= t)
                        {
                            antsList[k].curPos = antsList[k].toVisit[i];
                            antsList[k].visited.Add(antsList[k].toVisit[i]);
                            antsList[k].toVisit.Remove(antsList[k].toVisit[i]);
                            break;
                        }
                    }
                }

                antsList[k].distance = CalcDistance(antsList[k].visited, D);

                // Вычислить приращение
                for (int i = 1; i < antsList[k].visited.Count; i++)
                {
                    dteta[antsList[k].visited[i - 1], antsList[k].visited[i]] +=
Q/antsList[k].distance;
                    dteta[antsList[k].visited[i], antsList[k].visited[i - 1]] +=
Q/antsList[k].distance;
                }

                dteta[antsList[k].visited[antsList[k].visited.Count - 1],
antsList[k].visited[0]] += Q/antsList[k].distance;
                dteta[antsList[k].visited[0],
antsList[k].visited[antsList[k].visited.Count - 1]] += Q/antsList[k].distance;
            }

            // Найти минимальный L среди всех муравьёв
            for (int i = 0; i < antsList.Count; i++)
            {
                if (antsList[i].distance <= L)
                {
                    T.Clear();
                    L = antsList[i].distance;
                    T.AddRange(antsList[i].visited);
                }
            }
            PrintTL(T, L);

            // Изменение ферамонов
            for (int i = 0; i < eta.GetLength(0); i++)
                for (int j = 0; j < eta.GetLength(1); j++)
                    teta[i, j] = (1 - p) * teta[i, j] + dteta[i, j];
            PrintTeta(teta);

            antsList.Clear();
        }
        PrintTL(T, L);

        try
        {
```

```
                using (StreamWriter sw = new StreamWriter("graph.dot"))
                {
                    sw.Write(@"graph{ ");
                    for (int i = 0; i < D.GetLength(0); i++)
                        for (int j = 0; j < D.GetLength(1); j++)
                        {
                            if (i > j)
                                sw.WriteLine(@"{0} --
{1}[label=""{2:N1}"",len=""{2:0.000}""];", i, j, D[i, j]);
                        }

                    for (int i = 1; i < T.Count; i++)
                        sw.WriteLine(@"{0} --
{1}[label=""{2:N1}"",len=""{2:0.000}""][color=blue, penwidth=3.0];", T[i-1], T[i],
D[T[i - 1], T[i]]);
                    sw.WriteLine(@"{0} --
{1}[label=""{2:N1}"",len=""{2:0.000}""][color=blue, penwidth=3.0];", T[T.Count-1],
T[0], D[T[T.Count-1], T[0]]);
                    sw.Write(@"}");
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }

            string strCmdTxt;
            strCmdTxt = @""D:\Graphviz 2.28\bin\neato.exe"" -Tpng graph.dot -o
Граф.png";
            System.Diagnostics.Process.Start("CMD.exe", "/c " + strCmdTxt + " &
pause");

            Console.ReadKey();
        }

        private static void PrintTL(List<int> T, double L)
        {
            for (int i = 0; i < T.Count; i++)
                Console.Write("{0} ", T[i]);

            Console.WriteLine("\nDistance: {0}", L);
        }

        private static void PrintTeta(double[,] teta)
        {
            for (int i = 0; i < teta.GetLength(0); i++)
            {
                for (int j = 0; j < teta.GetLength(1); j++)
                    Console.Write("{0} ", teta[i, j]);
                Console.WriteLine();
            }
        }
    }
}
```

Тесты

Пусть β =1, ρ = 0.9

α	Расстояние
0	6538.93
0,2	4246.47
0,4	3709.38
0,6	3340.41
0,8	2938.36
1	2574.63
1,2	2836.14
1,4	2898.8

Теперь найдем такое β, при котором алгоритм работает эффективней всего.

α = 1, ρ = 0.9

β	Расстояние
0.5	3307.84
1	2574.63
2	2638.72
3	2611.08
4	2613.11
5	2721.69

Теперь найдем такое ρ, при котором алгоритм работает эффективней всего.

α = 1, β = 1

ρ	Расстояние
0.2	2713.43
0.4	2698.71
0.5	2685.32
0.8	2582.43
0.9	2574.63

Как мы можем видеть, эффективнее всего алгоритм работает при α = 1, β = 1, ρ = 0.9 Марко Дориго (изобретатель оптимизации по принципу муравьиной колонии) предлагает очень интересную дискуссию по параметрам алгоритма в статье «Система муравьев: оптимизация с помощью колонии сотрудничающих агентов». Был открыт ряд комбинаций α/β, которые позволяют находить хорошие решения за небольшое время.[1]

α	β
0.5	5
1	1
1	2
1	5

[1] Джонс, М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс .— 2011 .— С 82

Параметр α ассоциируется с количеством фермента, а параметр β – с видимостью (длинной грани). Чем больше значение параметра, тем он важнее для вероятностного уравнения, которое используется при выборе грани. Обратите внимание, что в одном случае значимость параметров равна. Во всех других случаях видимость более важна при выборе пути.
Как было показано на тестах данные типы параметров действительно соответствуют одним из самых лучших результатов. Также стоит сказать, что p следует выбирать в пределах от 0,5 до 1 не включая, тогда получаются наиболее приемлемые результаты. Другие значения p дают неудовлетворительные результаты.

Заключение

В проведённой работе осуществлена реализация муравьиного алгоритма применительно к задаче коммивояжёра. Сложность данного алгоритма, как несложно заметить, зависит от времени жизни колонии (t_{max}), количества городов (n) и количества муравьёв в колонии (m).

В качестве источников была использована статья Чуракова Михаила и Якушева Андрея “Муравьиные алгоритмы”