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

**Отчет**
По лабораторной работе №7
По курсу «Анализ Алгоритмов»

Тема: «Решение задачи коммивояжера»

Студент: Губайдуллин Р.Т.
Группа: ИУ7-51

Преподаватели: Волкова Л.Л.
Строганов Ю.В.

**Постановка задачи.**

Реализовать программу решения задачи коммивояжера используя алгоритм муравьиной колонии. Найти оптимальные параметры алгоритма, при которых задача решается относительно недолго.

**Алгоритм муравьиной колонии.**

*Начальная популяция.*

После создания популяции муравьев поровну распределяется по узлам сети. Необходимо равно распределить муравьев между узлами, чтобы все узлы имели одинаковые шансы стать отправной точкой. Если все муравьи начнут движение из одной точки, это будет означать, что данная точка является оптимальной для старта, а на самом деле мы этого не знаем.

*Движение муравья*

Движение муравья основывается на одном и очень простом вероятностном уравнении. Если муравей еще не закончил путь, то есть не посетил все узлы сети, для определения следующей грани пути используется уравнение:

$$\begin{cases} P_{ij,k}(t) = \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum\limits_{l \in J_{i,k}} \left[\tau_{il}(t)\right]^{\alpha} \cdot \left[\eta_{il}\right]^{\beta}}, j \in J_{i,k}; \\ P_{ij,k}(t) = 0, j \notin J_{i,k}; \end{cases}$$

Здесь τ – интенсивность фермента на грани между узлами i, j и η –функция, которая представляет измерение обратного расстояния для грани, α –вес фермента, а β – коэффициент эвристики. Параметры α и β определяют относительную значимость двух параметров, а также их влияние на уравнение.

*Путешествие муравья*

Пройденный муравьем путь отображается, когда муравей посетит все узлы диаграммы. Обратите внимание, что циклы запрещены, поскольку в алгоритм включен список табу. После завершения длина пути может быть подсчитана – она равна сумме всех граней, по которым путешествовал муравей.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}$$

Уравнение показывает количество фермента, который был оставлен на каждой грани(i,j) пути для муравья k.

Переменная Q является константой порядка всего пути.

*Испарение фермента*

В начале пути у каждой грани есть шанс быть выбранной. Чтобы постепенно удалить грани, которые входят в худшие пути в сети, ко всем граням применяется процедура испарения фермента (Pheromone evaporation). Используя константу ρ, мы получаем уравнение

$$\tau(t+1) = \tau(t)*(1-\rho)$$

Поэтому для испарения фермента используется коэффициент обновления феромона.

*Код программы*

```
public static void ant_colony_optimization(int[,] distance_matrix,double alpha,double beta,int e,int
Q,double p,int cities_number,int time_max, ref int length_route, ref List<int> shortest_route)
{
        //List<int> shortest_route = new List<int>();
        Random rnd = new Random();
        //int length_route = len;
        int length_to_town = 0;
        int current_town = 0;
        int time = 0;
        double[,] visibility = new double[cities_number,cities_number];
        double[,] pheromons = new double[cities_number, cities_number];

        for (int i = 0; i < cities_number; i++)
           for (int j = 0; j < cities_number; j++)
           {
              pheromons[i, j] = 0.5;
              visibility[i, j] = 1 / (distance_matrix[i, j] + 0.001);
           }

        while (time < time_max)
        {
           double[,] for_all_ants = new double[cities_number, cities_number];


           for (int k = 0; k < cities_number; k++)
           {
              length_to_town = 0; // Дистанция
              current_town = k;   // Текущий город
              List<int> ant_town = new List<int>();
              while (ant_town.Count < cities_number)
              {
                 List<int> desired_cities = new List<int>();
                 for (int tmp = 0; tmp < cities_number; tmp++)
                    desired_cities.Add(tmp);
                 foreach (var visited_town in ant_town)
                    desired_cities.Remove(visited_town);

                 double[] probability = new double[cities_number];
                 foreach(int j in desired_cities)
                 {
                    if (distance_matrix[current_town, j] != 0) // Если путь существует
                    {
                       double temp = 0;
                       foreach (int l in desired_cities)
                          temp += Math.Pow(pheromons[current_town, l], alpha) *
Math.Pow(visibility[current_town, l], beta);
                       probability[desired_cities.IndexOf(j)] =
                          Math.Pow(pheromons[current_town,j],alpha) *
Math.Pow(visibility[current_town,j],beta) / temp;
                    }
                    else
                       probability[desired_cities.IndexOf(j)] = 0;
                 }


                 double max_probability = probability.Max();
                if (max_probability.Equals(0))
                    break;

                 int selected_city = 0;
```

**Тесты**

Исходная матрица:

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 9 | 6 | 6 |
| 3 | 0 | 1 | 7 | 4 |
| 9 | 1 | 0 | 2 | 8 |
| 6 | 7 | 2 | 0 | 3 |
| 6 | 4 | 8 | 3 | 0 |

| Alpha | Beta | P | E | Time | Route | Len |
|---|---|---|---|---|---|---|
| 0,2 | 0 | 0,1 | 1 | 10 | 3,2,1,5,4 | 13 |
| 0,4 | 1 | 0,3 | 2 | 15 | 3,2,1,5,4 | 13 |
| 0,6 | 2 | 0,5 | 3 | 20 | 3,2,1,5,4 | 13 |
| 0,8 | 3 | 0,7 | 4 | 25 | 3,2,1,5,4 | 13 |
| 1 | 4 | 0,9 | 5 | 30 | 3,2,1,5,4 | 13 |

**Заключение**

В ходе лабораторной работы была реализована программа решения задачи коммивояжера с использованием алгоритма муравьиной колонии. Были найдены оптимальные параметры алгоритма, при которых задача решается относительно быстро.