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

ОТЧЕТ
По лабораторной работе №7
По курсу «Анализ алгоритмов»
Тема: **«Муравьиный алгоритм»**

Студент: Кононенко С. Д.
Группа: ИУ7-51

Москва, 2017

Постановка задачи

1. Реализовать муравьиный алгоритм на языке программирования
2. Сравнить работу алгоритма при разных значениях параметров задающих веса феромона и времени жизни колонии

Идея

Муравьиный алгоритм - один из эффективных полиномиальных алгоритмов для нахождения приближённых решений задачи коммивояжёра, а также решения аналогичных задач поиска маршрутов на графах. Суть подхода заключается в анализе и использовании модели поведения муравьёв, ищущих пути от колонии к источнику питания и представляет собой метаэвристическую оптимизацию.
С полным разбором алгоритма можно ознакомиться в книге М.В.Ульянова «Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ» [7.4]

Реализация

```
static void gogo_ant(ant_t *ant, matrix_t adj_mat, matrix_t weight, matrix_t
d_pheromon, int q)
{
    int N = weight.n;
    int i = 1;
    int next = 0;
    array_t prob = create_array(N);
    while (i < N)
    {

        float sum_weight = 0;
        for (int j = 0; j < N; j++)
            sum_weight += weight.matr[ant->curr_city][j] * ant->Jk.arr[j];
        for (int j = 0; j < N; j++)
        {
            prob.arr[j] = weight.matr[ant->curr_city][j] / sum_weight * ant-
>Jk.arr[j];
        }
        next = choose_next(prob);
        ant->curr_city = next;
        ant->Jk.arr[next] = 0;
        ant->route.arr[i++] = next;
    }
    ant->Lk = length_of_route(adj_mat, ant->route);
    add_pheromon(d_pheromon, ant->route, ant->Lk, q);
}

solution_t solve(matrix_t adj_mat, float a, float b, float p, int q, int
t_max)
{
    static int flag = 1;
    if (flag)
    {
        srand(time(NULL));
        flag = 0;
    }

    int N = adj_mat.n;

    ant_t *ants = create_ant_array(N); //array of ants, 1 per city;

    matrix_t pheromon = create_matrix(N, N);
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
```

```
            pheromon.matr[i][j] = 0.5;

    matrix_t visib = create_matrix(N, N);
    for (int i = 0; i < N; i++)
        for (int j = i; j < N; j++)
            if (i != j)
            {
                visib.matr[i][j] = 1 / adj_mat.matr[i][j];
                visib.matr[j][i] = visib.matr[i][j];
            }
            else
                visib.matr[i][j] = 666;

    matrix_t weight = create_matrix(N,N);
    recalc_weight(&weight, pheromon, visib, a, b);

    matrix_t d_pheromon = create_matrix(N, N);
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            d_pheromon.matr[i][j] = 0;

    int best_l = INT_MAX;
    array_t route = create_array(N);

    for (int t = 0; t < t_max; t++)
    {
        for (int k = 0; k < N; k++)
        {
            init_ant(&ants[k]);
            gogo_ant(&ants[k], adj_mat, weight, pheromon, q);
        }
        int best = -1;
        for (int i = 0; i < N; i++)
            if (ants[i].Lk < best_l)
            {
                best = i;
                best_l = ants[i].Lk;
            }
        if (best != -1)
            copy_array(route, ants[best].route);

        recalc_pheromon(&pheromon, &d_pheromon, p);
        recalc_weight(&weight, pheromon, visib, a, b);
    }

    solution_t solv = {best_l, route};
    free_ant_array(&ants, N);
    free_matrix(&pheromon);
    free_matrix(&weight);
    free_matrix(&visib);
    free_matrix(&d_pheromon);
    return solv;
}
```

recalc_weight() *и* ***recalc_pheromon()*** *функции осуществляющие перерасчет матрицы весов и матрицы феромонов на ребрах после каждой итерации*
choose_next() *– фунция осуществляющая случайный выбор пути на основе массива вероятностей*
length_of_route() *– функция считающая длину пройденного муравьем мути*
add_pheromon() *- функция изменяющая матрицу изменения кол-ва феромонов.*

Эксперимент

Параметры эксперимента:

-кол-во итераций 200

-коэфициент испарения 0.5

*-размерность матрицы смежностей 100*100*

-значения матрицы смежностей – случайные целые величины от 1 до 30

α	β	Длина найденного маршрута
0	1	534
0.1	0.9	534
0.2	0.8	449
0.3	0.7	401
0.4	0.6	391
0.5	0.5	388
0.6	0.4	372
0.7	0.3	371
0.8	0.2	392
0.9	0.1	569
1	0	1138

Таблица 1. Результаты эксперимента №1

Параметры эксперимента:

-коэффициент стадности(α) = 0.5

-коэффициент жадности(β) = 0.5

-коэфициент испарения 0.5

*-размерность матрицы смежностей 100*100*

-значения матрицы смежностей – случайные целые величины от 1 до 30

Итерации	Длина найденного маршрута
100	375
200	381
300	365
400	362
500	356
600	352
700	340
800	335
900	332
1000	321

Таблица 2. Результаты эксперимента №2

Вывод: 1. Оптимальным является соотношение *α = 0.6±0.1, β = 0.4±0.1 , при при других значения, алгоритм зачастую находит путь далекий от оптимального.*

2. Увеличение кол-ва итераций однозначно приводит к улучшению результата

Заключение

В ходе лабораторной работы был реализован муравьиный алгоритм,а также было проведено сравнение работы алгоритма при различных параметрах, задающих веса феромона и временя жизни колонии.