**Федеральное государственное бюджетное образовательное учреждение   
высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Анализ алгоритмов**

**Лабораторная работа №7**

**Отчёт на тему:**

# «Задача коммивояжёра»

Выполнила:

Янова Даниэлла

ИУ7-53

**Москва, 2018**

**Введение**

**Цель работы:** изучение муравьиного алгоритма для решения задачи коммивояжера, анализ влияния параметров алгоритма на время его выполнения.

1. **Аналитическая часть**

В данном разделе приведено описание алгоритма муравья.

**1.1 Постановка задачи коммивояжера**

**Дано:**

– множество городов, матрица – попарные расстояния между городами, .

**Найти**:

контур минимальной длины, цикл, проходящий через каждую вершину один раз и имеющий минимальный вес.

Данная задача может быть решена полным перебором всех вариантом. Но с увеличением числа городов количество маршрутов очень быстро возрастает, так как для n городов есть n! возможных маршрутов. Даже если учесть, что для одного контура есть два возможных направления и искать только одно из них, количество переборов снижается только до вариантов.

* 1. **Муравьиный алгоритм**

Муравьиный алгоритм – один из эффективных алгоритмов нахождения решений задачи коммивояжера. Алгоритм является жадным эвристическим алгоритмом: алгоритмом, принимающим локально оптимальные решения для каждой итерации.

Алгоритм основывается на поведении муравьев в поиске кратчайшего пути от колонии до источника питания. Моделирование поведения муравья связано с распределением феромонов на пути – на ребре в данном случае, количество феромона пропорционально длине маршрута. Таким образом, чем короче найденный маршрут, тем больше будет на нем феромонов. Чтобы алгоритм не сводился к единственному варианту, вводится обратная связь – «испарение» феромонов с пути со временем.

Муравьиный алгоритм использует три основных понятия – память муравья, видимость и феромонный след.

Память муравья – список всех вершин, которые он уже посетил. Так же есть список Ji, k – список вершин, которые нужно посетить i-тому муравью, находящемуся в вершине k.

Видимость – обратная расстоянию величина, , где Di,j – расстояние между вершинами i и j.

Феромонный след – «опыт» муравьев, коэффициент вероятности того, что муравей захочет перейти из вершины i в j – .

Вероятность перехода муравья k в вершину j из вершины i:

Если α = 0, то алгоритм становится «жадным», если β = 0, то муравьи оказываются «слепы» и опираются только на феромонный след.

В конце каждой итерации высчитывается общая протяженность маршрута и обнуляется память муравьев.

2. **Конструкторская часть**

В данном разделе представлен псевдокод алгоритма..

* 1. **Псевдокод алгоритма**

1. Ввод матрицы расстояний D
2. Инициализация параметров алгоритма
3. Инициализация рёбер – присвоение видимости ηi,j и начальной концентрации феромона
4. Размещение муравьёв
5. Выбор начального кратчайшего маршрута и определение L\*
6. Цикл по времени жизни колонии t=1, max t
7. Цикл по всем муравьям k=1,m
8. Построить маршрут Tk (t) рассчитать длину Lk (t)
9. Конец цикла по муравьям
10. Проверка всех L (t) на лучшее решение по сравнению сLk \*
11. В случае если решение L (t) лучше, обновить L\* и Tk \*
12. Цикл по всем рёбрам графа
13. Обновить следы феромона на ребр
14. Конец цикла по рёбрам
15. Конец цикла по времени
16. Вывести кратчайший маршрут T\* и его длину L\*

**3. Технологическая часть**

В данном разделе представлена реализация алгоритмов, указан язык программирования, а также необходимые модули.

**3.1 Средства реализации**

Для выполнения данной лабораторной работы использовался язык Python 3.7.1 в среде Pycharm. Замены времени проводились с использованием функции process\_time\_ns, входящей в библиотеку time Python версии 3.7.

**3.2 Листинг кода**

def aco(m, e, d, t\_max, alpha, beta, p, q):

nue = 1 / d

teta = np.random.sample((m, m))

T\_min = None

L\_min = None

t = 0

while t < t\_max:

teta\_k = np.zeros((m, m))

for k in range(m):

Tk = [k]

Lk = 0

i = k

while len(Tk) != m:

J = [r for r in range(m)]

for c in Tk:

J.remove(c)

P = [0 for alpha in J]

for j in J:

if d[i][j] != 0:

buf = sum((teta[i][l] \*\* alpha) \* (nue[i][l] \*\* beta) for l in J)

P[J.index(j)] = (teta[i][j] \*\* alpha) \* (nue[i][j] \*\* beta) / buf

else:

P[J.index(j)] = 0

Pmax = max(P)

if Pmax == 0:

break

index = P.index(Pmax)

Tk.append(J[index])

Lk += d[i][J[index]]

i = J.pop(index)

if L\_min is None or (Lk + d[Tk[0]][Tk[-1]]) < L\_min:

L\_min = Lk + d[Tk[0]][Tk[-1]]

T\_min = Tk

for g in range(len(Tk) - 1):

alpha= Tk[g]

betha = Tk[g + 1]

teta\_k[alpha][betha] += q / Lk

teta\_e = (e \* q / L\_min) if L\_min else 0

teta = (1 - p) \* teta + teta\_k + teta\_e

t += 1

return L\_min

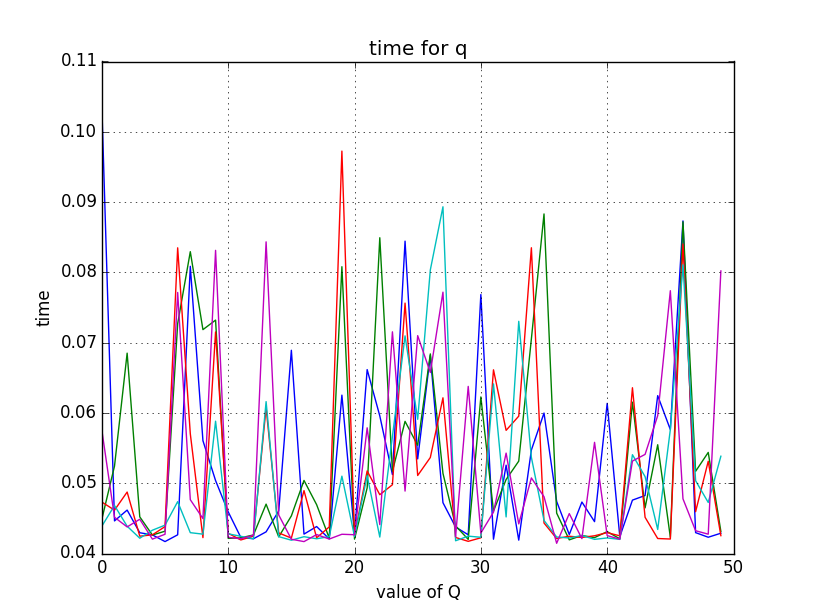
**Листинг 1.1 – Реализация муравьиного алгоритма**

1. **Экспериментальная часть**

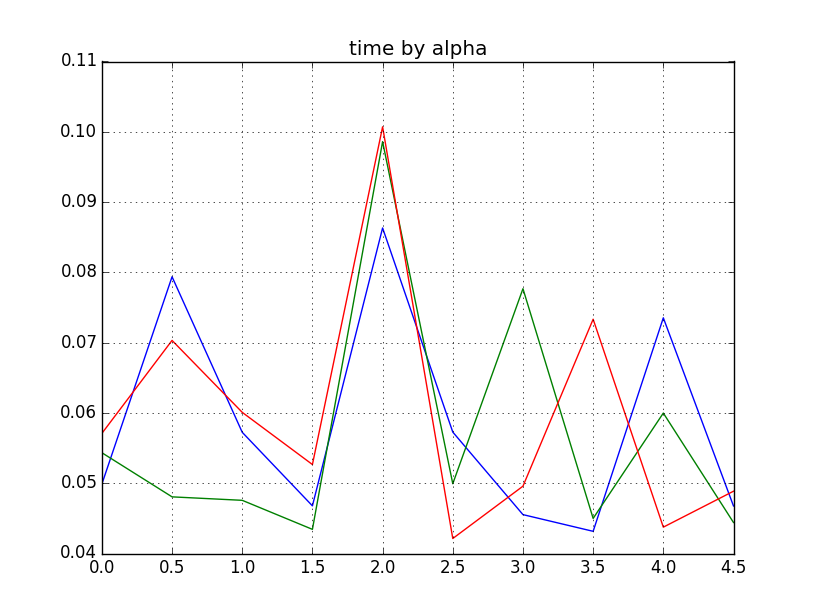
В данном разделе будут произведены замеры времени работы алгоритма. Для исследования скоростных характеристик был использован компьютер на базе процессора Intel Core i7-6600U, содержащий 16 гигабайт оперативной памяти. Модуль тестирования запускался с жестокого диска под операционной системой Windows. Жесткий диск имел среднюю скорость передачи данных при чтении 2629 Мбайт/с и 798 Мбайт/с при записи.

**4.1 Тестирование алгоритма**

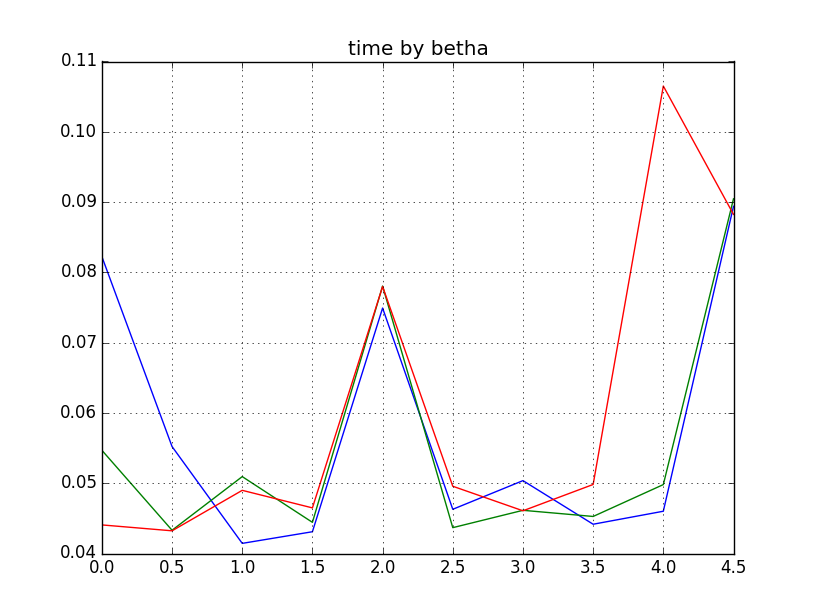
Для проверки значимости Q – параметра, имеющего значение порядка длины оптимального пути, зафиксируем значения α, β и p и будем изменять Q в диапазоне [0, 50] с шагом 1.

Рисунок 4.1 – Влияние параметра Q на время работы алгоритма

Для проверки значимости α, зафиксируем Q = 1, β = 0 и p = 0.5 и будем изменять α в диапазоне [0, 5] с шагом 0.5.

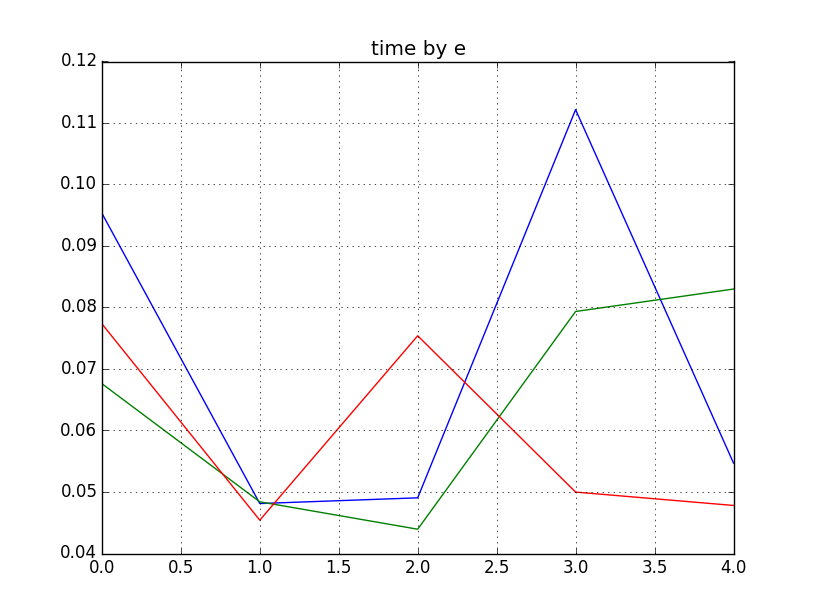
**Рисунок 4.2 –** Влияние параметра α на время работы алгоритма

Для проверки значимости β, зафиксируем Q = 1, α = 0 и p = 0.5 и будем изменять β в диапазоне [0, 5] с шагом 0.5.

**Рисунок 4.3 –** Влияние параметра β на время работы алгоритма

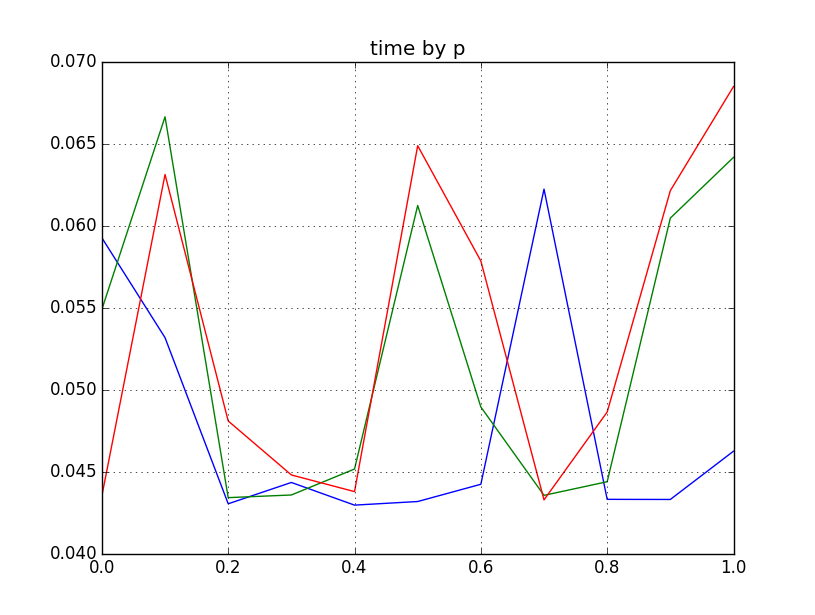
Как видно из рисунков 4.2 и 4.3 у параметров α и β есть «нежелательное» значение, равное двум, на котором алгоритм работает неэффективно.

Для проверки значимости e, зафиксируем Q = 1, α = 1, β = 1 и p = 0.5 и будем изменять e в диапазоне [0, 5] с шагом 0.5.

Рисунок 4.4 – Влияние параметра e на время работы алгоритма

Как видно из рисунка 4.4, лучшим значением параметра e является единица, т. е. оптимальной ситуацией является та, где в колонии есть один «элитный» муравей.

Для проверки значимости p, зафиксируем Q = 1, α = 1, β = 1 и e =1 и будем изменять e в диапазоне [0, 1] с шагом 0.1.

Рисунок 4.5 – Влияние параметра p на время работы алгоритма

Как видно из рисунка 4.5, лучшим значением параметра p является 0.3.

**Заключение**

В данной работе был изучен муравьиный алгоритм для решения задачи коммивояжера и проанализировано влияние параметров алгоритма на время его выполнения.