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 для задачи коммивояжера»

Студент: Жарова Е.А.

Группа ИУ7-51

Москва, 2017

Оглавление

Постановка задачи

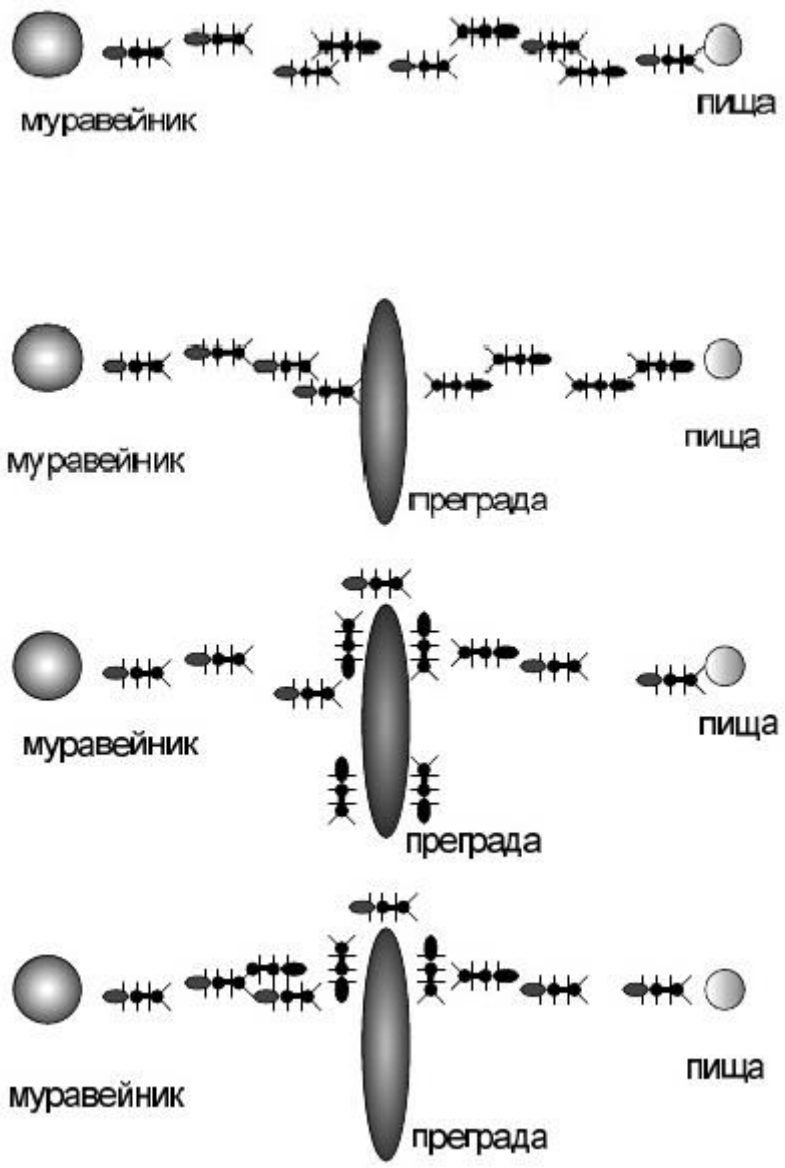
Необходимо изучить и реализовать муравьиный алгоритм для задачи коммивояжера.

Описание алгоритма

Муравьиные алгоритмы представляют собой вероятностную жадную эвристику, где вероятности устанавливаются, исходя из информации о качестве решения, полученной из предыдущих решений. Они могут использоваться как для статических, так и для динамических комбинаторных оптимизационных задач. Сходимость гарантирована, то есть в любом случае мы получим оптимальное решение, однако скорость сходимости неизвестна

Концепция муравьиных алгоритмов

Идея муравьиного алгоритма – моделирование поведения муравьёв, связанного с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый кратчайший путь. При своём движении муравей метит путь феромоном, и эта информация используется другими муравьями для выбора пути. Это элементарное правило поведения и определяет способность муравьёв находить новый путь, если старый оказывается недоступным. Рассмотрим случай, показанный на рисунке, когда на оптимальном доселе пути возникает преграда. В этом случае необходимо определение нового оптимального пути. Дойдя до преграды, муравьи с равной вероятностью будут обходить её справа и слева. То же самое будет происходить и на обратной стороне преграды. Однако, те муравьи, которые случайно выберут кратчайший путь, будут быстрее его проходить, и за несколько передвижений он будет более обогащён феромоном. Поскольку движение муравьёв определяется концентрацией феромона, то следующие будут предпочитать именно этот путь, продолжая обогащать его феромоном до тех пор, пока этот путь по какой-либо причине не станет недоступен.



Рассмотрим случай, показанный на рисунке, когда на оптимальном доселе пути возникает преграда. В этом случае необходимо определение нового оптимального пути. Дойдя до преграды, муравьи с равной вероятностью будут обходить её справа и слева. То же самое будет происходить и на обратной стороне преграды. Однако, те муравьи, которые случайно выберут кратчайший путь, будут быстрее его проходить, и за несколько передвижений он будет более обогащён феромоном. Поскольку движение муравьёв определяется концентрацией феромона, то следующие будут предпочитать именно этот путь, продолжая обогащать его феромоном до тех пор, пока этот путь по какой-либо причине не станет недоступен.

Очевидная положительная обратная связь быстро приведёт к тому, что кратчайший путь станет единственным маршрутом движения большинства муравьёв. Моделирование испарения феромона – отрицательной обратной связи – гарантирует нам, что найденное локально оптимальное решение не будет единственным – муравьи будут искать и другие пути. Если мы моделируем процесс такого поведения на некотором графе, рёбра которого представляют собой возможные пути перемещения муравьёв, в течение определённого времени, то наиболее обогащённый феромоном путь по рёбрам этого графа и будет являться решением задачи, полученным с помощью муравьиного алгоритма.

Реализация муравьиного алгоритма для задачи коммивояжера

Задача формулируется как задача поиска минимального по стоимости замкнутого маршрута по всем вершинам без повторений на полном взвешенном графе с n вершинами. Содержательно вершины графа являются городами, которые должен посетить коммивояжёр, а веса рёбер отражают расстояния (длины) или стоимости проезда. Эта задача является NP-трудной, и точный переборный алгоритм её решения имеет факториальную сложность. Моделирование поведения муравьёв связано с распределением феромона на тропе – ребре графа в задаче коммивояжёра. При этом вероятность включения ребра в маршрут отдельного муравья пропорциональна количеству феромона на этом ребре, а количество откладываемого феромона пропорционально длине маршрута. Чем короче маршрут, тем больше феромона будет отложено на его рёбрах,

следовательно, большее количество муравьёв будет включать его в синтез собственных маршрутов.

Моделирование такого подхода, использующего только положительную обратную связь, приводит к преждевременной сходимости – большинство муравьёв двигается по локально оптимальному маршруту. Избежать этого можно, моделируя отрицательную обратную связь в виде испарения феромона. При этом если феромон испаряется быстро, то это приводит к потере памяти колонии и забыванию хороших решений, с другой стороны, большое время испарения может привести к получению устойчивого локального оптимального решения.

Теперь с учётом особенностей задачи коммивояжёра, мы можем описать локальные правила поведения муравьёв при выборе пути

1. Муравьи имеют собственную «память». Поскольку каждый город может быть посещён только один раз, то у каждого муравья есть список уже посещённых городов – список запретов. Обозначим через список городов, которые необходимо посетить муравью *k*, находящемуся в городе $J_{i,k}$ список городов, которые необходимо посетить муравью *k*, находящемуся в городе *i*.

2. Муравьи обладают «зрением» – видимость есть эвристическое желание посетить город *j*, если муравей находится в городе i. Будем считать, что видимость обратно пропорциональна расстоянию между городами $\eta = 1 / D_{ij}$.

3. Муравьи обладают «обонянием» – они могут улавливать след феромона, подтверждающий желание посетить город *j* из города *i* на основании опыта других муравьёв. Количество феромона на ребре *(i,j)* в момент времени *t* обозначим через $\tau_{ij}(t)$

4. На этом основании мы можем сформулировать вероятностно-пропорциональное правило, определяющее вероятность перехода k-ого муравья из города i в город j:

$$\begin{cases} P_{ij,k}(t) = \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in J_{i,k}} \left[\tau_{il}(t)\right]^{\alpha} \cdot \left[\eta_{il}\right]^{\beta}}, j \in J_{i,k}; \\ P_{ij,k}(t) = 0, j \notin J_{i,k}; \end{cases} \quad (1)$$

   Где α, β - параметры, задающие веса следа феромона. При α = 0 алгоритм вырождается до жадного алгоритма (будет выбран ближайший город). Заметим, что выбор города является вероятностным, правило (1) лишь определяет ширину зоны города *j*; в общую зону всех городов $J_{i,k}$ бросается случайное число, которое и определяет выбор муравья. Правило (1) не изменяется в ходе алгоритма, но у двух разных муравьёв значение вероятности перехода будут отличаться, т.к. они имеют разный список разрешённых городов.

5. Пройдя ребро *(i,j)*, муравей откладывает на нём некоторое количество феромона, которое должно быть связано с оптимальностью сделанного выбора. Пусть $T_k(t)$ есть маршрут, пройденный муравьём *k* к моменту времени *t*, $L_k(t)$ – длина этого маршрута, а *Q* – параметр, имеющий значение порядка длины оптимального пути. Тогда откладываемое количество феромона может быть задано в виде

$$\Delta\tau_{ij,k}(t)=\begin{cases}\dfrac{Q}{L_k(t)},(i,j)\in T_k(t);\\ 0,(i,j)\notin T_k(t);\end{cases}$$

Правила внешней среды определяют, в первую очередь, испарение феромона. Пусть ρ ∈ [0,1], есть коэффициент испарения, тогда правило испарения имеет вид

$$\tau_{ij}(t+1)=(1-p)\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t);\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij,k}(t), \qquad (2)$$

Где m – количество муравьев в колонии.

В начале алгоритма количества феромона на рёбрах принимается равным небольшому положительному числу. Общее количество муравьёв остаётся постоянным и равным количеству городов, каждый муравей начинает маршрут из своего города.

Дополнительная модификация алгоритма может состоять в ведении так называемых «элитных» муравьёв, которые усиливают рёбра наилучшего маршрута, найденного с начала работы алгоритма. Обозначим через T^* наилучший текущий маршрут, через L^* – его длину. Тогда если в колонии есть e элитных муравьёв, то рёбра маршрута получат дополнительное количество феромона

$$\Delta\tau_e=e\cdot Q/L^* \qquad (3)$$

Муравьиный алгоритм для задачи коммивояжера в псевдокоде

1. Ввод матрицы расстояний D
2. Инициализация параметров алгоритма - α,β,e,Q
3. Инициализация рёбер - присвоение видимости η_{ij} 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 $t=1,t_{max}$
7. Цикл по всем муравьям $k=1,m$
8. Построить маршрут $T_k(t)$ по правилу (1) и рассчитать длину $L_k(t)$
9. конец цикла по муравьям
10. Проверка всех $L_k(t)$ на лучшее решение по сравнению с L^*
11. В случае если решение $L_k(t)$ лучше, обновить L^* и T^*
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^*

Сложность данного алгоритма, как несложно заметить, зависит от времени жизни колонии (t_{max}), количества городов (n) и количества муравьёв в колонии (m).

Реализация

Алгоритм был реализован на языке Python.

```
from random import randint
import numpy
from time import time
#shortest_route - длина этого маршрута


def create_distance_matrix(cities_number):
    matrix = numpy.zeros((cities_number, cities_number))
    max_len = int(input('Введите максимальное расстояние '))
    for i in range(cities_number):
        for j in range(i + 1, cities_number):
            distance = randint(1, max_len)  # от min до max расстояния
            matrix[i][j], matrix[j][i] = distance, distance # для того, чтобы
матрица расстояний была симметричной

    return matrix


def ant_colony_optimization(distance_matrix, alpha, beta, e, Q, p,
cities_number,max_time_of_life):
    shortest_route, length_route = None, None  # кратчайший путь и его длина
    pheromons = numpy.random.sample((cities_number, cities_number))#
размещение
    visibility = 1 /( distance_matrix+0.0001)# присовение видимости

    time_of_life = 0
    begin_time = time()
    while time_of_life < max_time_of_life:#6
        for_all_ants = numpy.zeros((cities_number, cities_number)) #генерация

        # для каждого муравья:
        for k in range(cities_number):
            ant_town, length_to_town = [k], 0
            current_town = k
            while len(ant_town) != cities_number:
                desired_cities = [r for r in range(cities_number)]
                for visited_town in ant_town:
                    desired_cities.remove(visited_town)
                probability = [0 for t in desired_cities]
                for j in desired_cities:
                    if distance_matrix[current_town][j] != 0:
                        temp = sum(
                            (pheromons[current_town][l] ** alpha) *
(visibility[current_town][l] ** beta) for l in
                            desired_cities)
                        probability[desired_cities.index(j)] = \
                            (pheromons[current_town][j] ** alpha) *
(visibility[current_town][j] ** beta) / temp
                    else:
                        probability[desired_cities.index(j)] = 0

                max_probability = max(probability)
                if max_probability == 0: # проверка на лучшее решение (10)
                    return "Муравьишка в беде, он изолирован!"

                selected_city = probability.index(max_probability)
                ant_town.append(desired_cities[selected_city])
                length_to_town +=
distance_matrix[current_town][desired_cities[selected_city]]
                current_town = desired_cities.pop(selected_city)
```

```
            if length_route is None or (length_to_town +
distance_matrix[ant_town[0]][ant_town[-1]]) < length_route:
                length_route = length_to_town +
distance_matrix[ant_town[0]][ant_town[-1]]
                shortest_route = ant_town

            for pheromone in range(len(ant_town) - 1): # -1 из-за
особенностей python (12)
                a = ant_town[pheromone]
                b = ant_town[pheromone + 1]
                for_all_ants[a][b] += Q / length_to_town

        for_elite = (e * Q / length_route) if length_route else 0
        pheromons = (1 - p) * pheromons + for_all_ants + for_elite
        time_of_life += 1
    length_route=0
    end_time = time()
    for i in range(len(distance_matrix)-1):
        length_route+=distance_matrix[shortest_route[i]][shortest_route[i+1]]
    #print(end_time-begin_time)

    return shortest_route, length_route

cities_number=int(input('Введите количество городов '))
matrix=create_distance_matrix(cities_number)
print(matrix)
matrix_beta = matrix
#alpha = float(input('Введите вес ферамона alpha '))  # вес феромона
#beta = float(input('Введите вес ферамона beta '))  # вес феромона
#e = int(input('Введите количеств ферамона, выделяемое элитным муравьем ')) #
элитные муравьи
#Q = int(input('Порядок цены оптимального решения '))  # параметр, имеющий
значение порядка цены оптимального решения
#p = float(input('Ваедите интенсивность испарения ')) # интенсивность
испарения
while(1):
    #print('\n')
    max_time_of_life=(int(input('Введите количество поколений муравьев  ')))
    #max_time_of_life = 100
    #alpha = float(input('Введите вес ферамона alpha '))  # вес феромона
    beta = float(input('Введите вес ферамона beta '))  # вес феромона
    #e = int(input('Введите количеств ферамона, выделяемое элитным муравьем
')) # элитные муравьи
    #Q = int(input('Порядок цены оптимального решения '))  # параметр,
имеющий значение порядка цены оптимального решения
    #p = float(input('Ваедите интенсивность испарения ')) # интенсивность
испарения
    #alpha = 1
    alpha = 1-beta
    e = 3
    Q = 10
    p = 0.5

    #print('\n')

print(ant_colony_optimization(matrix,alpha,beta,e,Q,p,cities_number,max_time_
of_life))
```

Примеры работы

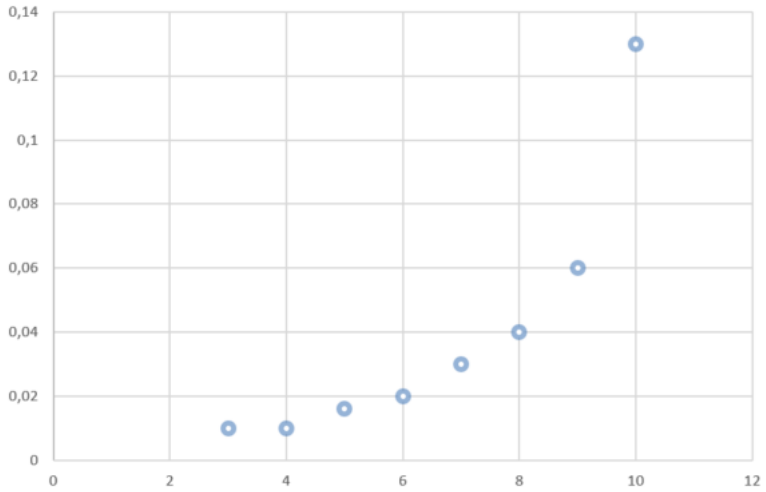
```
Введите количество городов 5
Введите максимальное расстояние 10
[[0. 2. 7. 5. 9.]
 [2. 0. 7. 7. 8.]
 [7. 7. 0. 9. 7.]
 [5. 7. 9. 0. 9.]
 [9. 8. 7. 9. 0.]]


Введите количество поколений муравьев  10


([1, 0, 3, 2, 4], 23.0)


Введите количество поколений муравьев  100


([2, 1, 0, 3, 4], 23.0)


Введите количество поколений муравьев  1000


([3, 4, 2, 1, 0], 25.0)


Введите количество поколений муравьев  10000


([0, 1, 2, 4, 3], 25.0)
```

Исследования

Для исследования быстродействия программы были выбраны следующие параметры:

Количество городов	Время работы, мс
3	0.009936094
4	0.009983063
5	0.015994072
6	0.020020962
7	0.029985905
8	0.039920092
9	0.059988976
10	0.009936094

Отобразим полученные результаты на графике:



Зависимость результатов от alpha

В данном исследовании alpha при прочих неизменных коэффициентах (коэффициент испарения 0.5, 10 городов, alpha -задается с клавиатуры, а beta вычисляется по формуле 1 - alpha)

```
Введите количество городов 10
Введите максимальное расстояние 10
[[ 0.  5. 10.  8.  1.  9.  6.  3.  4.  9.]
 [ 5.  0.  8.  5. 10.  1.  6.  5.  7.  8.]
 [10.  8.  0.  8.  9.  2.  2. 10. 10.  9.]
 [ 8.  5.  8.  0.  6.  4.  2.  5.  4.  3.]
 [ 1. 10.  9.  6.  0.  6.  8.  6.  5.  5.]
 [ 9.  1.  2.  4.  6.  0.  6.  7.  4. 10.]
 [ 6.  6.  2.  2.  8.  6.  0.  4.  8. 10.]
 [ 3.  5. 10.  5.  6.  7.  4.  0.  2.  7.]
 [ 4.  7. 10.  4.  5.  4.  8.  2.  0.  7.]
 [ 9.  8.  9.  3.  5. 10. 10.  7.  7.  0.]]
Введите вес ферамона alpha 1
([2, 5, 3, 1, 4, 8, 0, 9, 7, 6], 50.0)
Введите вес ферамона alpha 0.9
([4, 5, 1, 2, 6, 0, 7, 8, 3, 9], 35.0)
Введите вес ферамона alpha 0.8
([1, 5, 8, 0, 4, 7, 6, 2, 9, 3], 34.0)
Введите вес ферамона alpha 0.7
([4, 0, 1, 5, 2, 6, 3, 9, 7, 8], 25.0)
Введите вес ферамона alpha 0.6
([1, 5, 2, 6, 3, 9, 4, 0, 7, 8], 21.0)
Введите вес ферамона alpha 0.5
([0, 4, 9, 3, 6, 2, 5, 1, 7, 8], 23.0)
Введите вес ферамона alpha 0.4
([1, 5, 2, 6, 3, 9, 4, 0, 7, 8], 21.0)
Введите вес ферамона alpha 0.3
([1, 5, 2, 6, 3, 9, 4, 0, 8, 7], 22.0)
Введите вес ферамона alpha 0.2
([0, 4, 9, 3, 6, 2, 5, 1, 7, 8], 23.0)
Введите вес ферамона alpha 0.1
([2, 5, 1, 7, 8, 0, 4, 9, 3, 6], 25.0)
Введите вес ферамона alpha 0
([7, 8, 0, 4, 9, 3, 6, 2, 5, 1], 22.0)
```

Вывод: при низком значении alpha не реализуется преимущество феромонов. После прохождения некоторого оптимального значения феромоны сводят задачу к субоптимальному решению, не позволяя получить более приемлемое значение.

Зависимость результатов от beta

В данном исследовании эксперимент проводился так же, как и в предыдущий, но применительно к beta.

```
Введите количество городов 10
Введите максимальное расстояние 10
[[ 0. 10.  6.  8.  9.  4.  4.  7.  8.  3.]
 [10.  0.  3.  5.  8. 10.  7. 10.  5.  4.]
 [ 6.  3.  0.  5. 10.  6.  3.  3.  7.  1.]
 [ 8.  5.  5.  0.  8.  2.  9.  8.  6.  5.]
 [ 9.  8. 10.  8.  0.  9.  1.  7.  9.  3.]
 [ 4. 10.  6.  2.  9.  0.  2.  8.  8.  1.]
 [ 4.  7.  3.  9.  1.  2.  0.  2.  7.  7.]
 [ 7. 10.  3.  8.  7.  8.  2.  0.  3.  3.]
 [ 8.  5.  7.  6.  9.  8.  7.  3.  0.  2.]
 [ 3.  4.  1.  5.  3.  1.  7.  3.  2.  0.]]
Введите вес ферамона beta 1
([7, 6, 4, 9, 2, 1, 3, 5, 0, 8], 29.0)
Введите вес ферамона beta 0.9
([0, 9, 5, 3, 2, 1, 8, 7, 6, 4], 25.0)
Введите вес ферамона beta 0.8
([0, 9, 2, 1, 8, 7, 6, 4, 3, 5], 28.0)
Введите вес ферамона beta 0.7
([0, 6, 4, 9, 5, 3, 2, 1, 8, 7], 27.0)
Введите вес ферамона beta 0.6
([4, 6, 7, 8, 9, 2, 1, 3, 5, 0], 23.0)
Введите вес ферамона beta 0.5
([7, 6, 4, 9, 0, 5, 3, 8, 1, 2], 29.0)
Введите вес ферамона beta 0.4
([6, 4, 9, 2, 1, 3, 5, 0, 8, 7], 30.0)
Введите вес ферамона beta 0.3
([7, 6, 4, 9, 2, 1, 3, 5, 0, 8], 29.0)
Введите вес ферамона beta 0.2
([2, 6, 4, 9, 8, 0, 5, 3, 1, 7], 38.0)
Введите вес ферамона beta 0.1
([6, 0, 7, 2, 5, 3, 1, 8, 9, 4], 37.0)
Введите вес ферамона beta 0
([5, 0, 7, 8, 4, 9, 3, 1, 2, 6], 42.0)
```

Вывод:

При высоких значениях beta алгоритм решения сводится к жадному, что не позволяет добиться оптимального результата.

Зависимость результатов от скорости испарения

```
Введите количество городов 10
Введите максимальное расстояние 10
[[ 0.  9.  4.  5.  7.  6.  5.  4. 10.  5.]
 [ 9.  0. 10.  5.  5. 10.  4.  8.  4.  1.]
 [ 4. 10.  0.  8.  7. 10.  2. 10.  6.  6.]
 [ 5.  5.  8.  0.  8.  4.  5.  1.  2.  7.]
 [ 7.  5.  7.  8.  0.  3.  1.  1.  7.  5.]
 [ 6. 10. 10.  4.  3.  0.  8.  1.  6.  8.]
 [ 5.  4.  2.  5.  1.  8.  0.  2.  8.  5.]
 [ 4.  8. 10.  1.  1.  1.  2.  0.  7.  4.]
 [10.  4.  6.  2.  7.  6.  8.  7.  0.  8.]
 [ 5.  1.  6.  7.  5.  8.  5.  4.  8.  0.]]
Ваедите интенсивность испарения 0.1
([2, 0, 5, 7, 3, 8, 1, 9, 4, 6], 25.0)
Ваедите интенсивность испарения 0.2
([3, 7, 5, 4, 6, 2, 0, 9, 1, 8], 22.0)
Ваедите интенсивность испарения 0.3
([9, 1, 8, 3, 7, 5, 4, 6, 2, 0], 19.0)
Ваедите интенсивность испарения 0.4
([9, 1, 8, 3, 7, 5, 4, 6, 2, 0], 19.0)
Ваедите интенсивность испарения 0.5
([4, 6, 2, 0, 9, 1, 8, 3, 7, 5], 21.0)
Ваедите интенсивность испарения 0.6
([2, 6, 7, 4, 5, 3, 8, 1, 9, 0], 24.0)
Ваедите интенсивность испарения 0.7
([2, 6, 4, 7, 5, 3, 8, 1, 9, 0], 21.0)
Ваедите интенсивность испарения 0.8
([8, 3, 7, 5, 4, 6, 2, 0, 9, 1], 20.0)
Ваедите интенсивность испарения 0.9
([5, 7, 4, 6, 2, 0, 9, 1, 8, 3], 21.0)
Ваедите интенсивность испарения 1
([0, 9, 1, 8, 3, 7, 5, 4, 6, 2], 20.0)
```

Вывод: с увеличением количества элитных муравьев сначала следует улучшение результата, а затем, как и в предыдущих случаях, это не позволяет добиться оптимального результата.

Сводная таблица

alpha	beta	p	L_{min}
1	0	0	41
0,9	0,1	0,1	34
0,8	0,2	0,2	29
0,7	0,3	0,3	21
0,6	0,4	0,4	22
0,5	0,5	0,5	20
0,4	0,6	0,6	22
0,3	0,7	0,7	20
0,2	0,8	0,8	22
0,1	0,9	0,9	20
0	1	1	20

Заключение

Был изучен и реализован муравьиный алгоритм для задачи коммивояжера. Была получена экспоненциальная зависимость времени работы алгоритма от количества вершин графа.

Было проведено исследование зависимости оптимальности результата от параметров уравнения. Были получен следующий вывод:

Для каждого коэффициента целесообразно задать ряд значений и сравнить результаты работы программы при каждом наборе. Заранее определить оптимальные величины коэффициентов не представляется возможным, однако, точно можно сказать, что alpha и beta должны быть одного порядка и прочие коэффициенты не должны обращаться в ноль или принимать очевидно большие или малые значения.