Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа №7
по дисциплине "Анализ Алгоритмов"

Тема Муравьиный алгоритм

Студент Светличная А.А.

Группа ИУ7-53Б

Преподаватель Волкова Л. Л., Строганов Ю.В.

Москва — 2022 г.

Оглавление

Введение

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.

1 Аналитическая часть

1.1 Цель и задачи

Целью данной лабораторной работы является изучение муравьиного алгоритма на примере задачи коммивояжера.

Для достижения поставленной цели необходимо выполнить следующие **задачи**:

1) исследовать задачу коммивояжера;
2) изучить алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжера;
3) провести параметризацию муравьиного алгоритма на нескольких классах данных;
4) составить схемы используемых алгоритмов;
5) реализовать алгоритмы полного перебора и муравьиный алгоритм;
6) провести сравнительный анализ времени работы данных алгоритмов;
7) описать и обосновать полученные результаты в отчете.

1.2 Задача коммивояжера

В задаче коммивояжера рассматривается n городов и матрица попарно различных расстояний между ними. Требуется найти такой порядок посещения городов, чтобы суммарное пройденное расстояние было минимальным, каждый город посещался ровно один раз. Иногда условием задачи коммивояжера является возврат в тот город, с которого начинался маршрут.

1.3 Алгоритм полного перебора

Для решения задачи коммивояжера алгоритм полного перебора предполагает рассмотрение всех возможных путей в графе и выбор наименьшего из них. Смысл перебора состоит в том, что перебираются все варианты объезда городов и выбирается оптимальный, что гарантирует точное решение задачи. Однако, при таком подходе количество возможных маршрутов очень быстро возрастает с ростом n (сложность алгоритма равна $n!$).

1.4 Муравьиный алгоритм

Муравьиный алгоритм [1] — метод решения задач коммивояжера на основании моделирования поведения колонии муравьев.

Каждый муравей определяет для себя маршрут, который необходимо пройти на основе феромона, который он ощущает во время прохождения, каждый муравей оставляет феромон на своем пути, чтобы остальные муравьи могли по нему ориентироваться. В результате при прохождении каждым муравьем различного маршрута наибольшее число феромона остается на оптимальном пути.

Для каждого муравья переход из города i в город j зависит от трех составляющих: памяти муравья, видимости и виртуального следа феромона.

- Память муравья — это список посещенных муравьем городов, заходить в которые еще раз нельзя. Используя этот список, муравей гарантированно не попадет в один и тот же город дважды. Данный список возрастает при совершении маршрута и обнуляется в начале каждой итерации алгоритма.

- Видимость — величина, обратная расстоянию, рассчитывающаяся по формуле 1.1.

$$\eta_{ij} = 1/D_{ij}, \tag{1.1}$$

  где D_{ij} — расстояние между городами i и j. Видимость — это локальная статическая информация, выражающая эвристическое желание

посетить город j из города i, то есть чем ближе город, тем больше желание посетить его.

- Виртуальный след феромона на ребре (i, j) представляет подтвержденное муравьиным опытом желание посетить город j из города i. След феромона является глобальной и динамичной информацией — она изменяется после каждой итерации алгоритма, отражая приобретенный муравьями опыт.

Формула вычисления вероятности перехода в заданную точку (1.2).

$$P_{kij} = \begin{cases} \frac{\tau_{ij}^a \eta_{ij}^b}{\sum_{q=1}^{m} \tau_{iq}^a \eta_{iq}^b}, \text{вершина не была посещена ранее муравьем k,} \\ 0, \text{иначе,} \end{cases} \tag{1.2}$$

где a — параметр влияния длины пути, b — параметр влияния феромона, τ_{ij} — расстояния от города i до j, η_{ij} — количество феромонов на ребре (i, j).

Правило обновления феромона после движения всех муравьев:

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij}. \tag{1.3}$$

При этом

$$\Delta\tau_{ij} = \sum_{k=1}^{N} \tau_{ij}^k, \tag{1.4}$$

где

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k, \text{ребро посещено k-ым муравьем,} \\ 0, \text{иначе.} \end{cases} \tag{1.5}$$

Существует несколько оптимизаций данного алгоритма, одна из которых введение так называемых элитных муравьев [2]. Элитный муравей усиливает ребра наилучшего маршрута, найденного с начала работы алгоритма. Количество феромона, откладываемого на ребрах наилучшего текущего маршрута T^+, принимается равным Q/L^+, где L^+ — длина маршрута T^+. Этот феромон побуждает муравьев к исследованию решений, содержащих несколько ребер наилучшего на данный момент маршрута T^+. Если в

муравейнике есть e элитных муравьев, то ребра маршрута T^+ будут получать общее усиление:

$$\Delta\tau_e = e * Q/L^+. \tag{1.6}$$

Вывод

В этом разделе была изучена задача коммивояжера и используемые для её решения алгоритмы: полный перебор и муравьиный алгоритм с оптимизацией в виде элитных муравьев.

2 Конструкторская часть

Описание алгоритмов

На рисунке 2.1, 2.2 представлены схемы алгоритма полного перебора и муравьиного алгоритма соответсвенно.

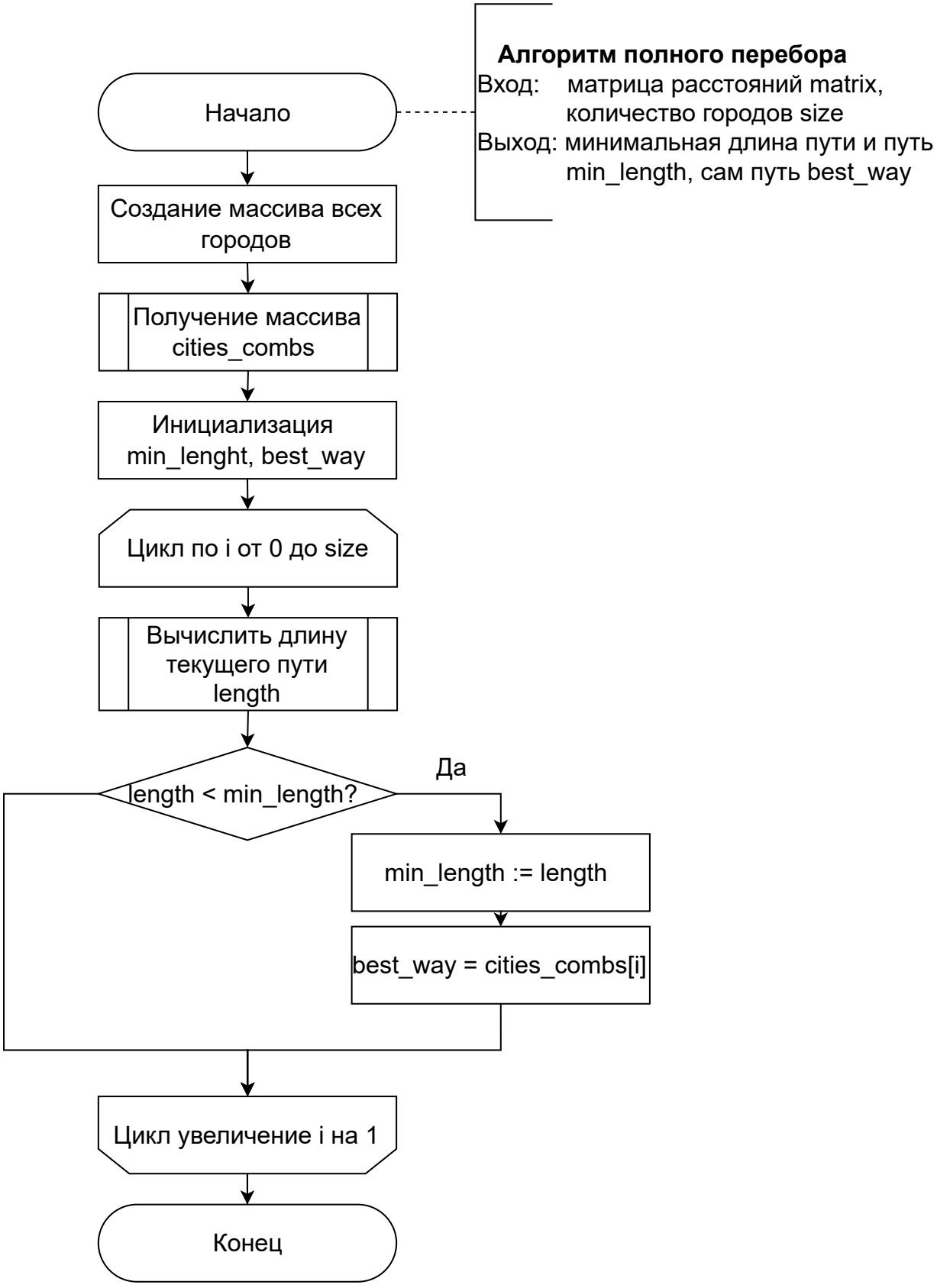

Рисунок 2.1 – Схема алгоритма полного перебора

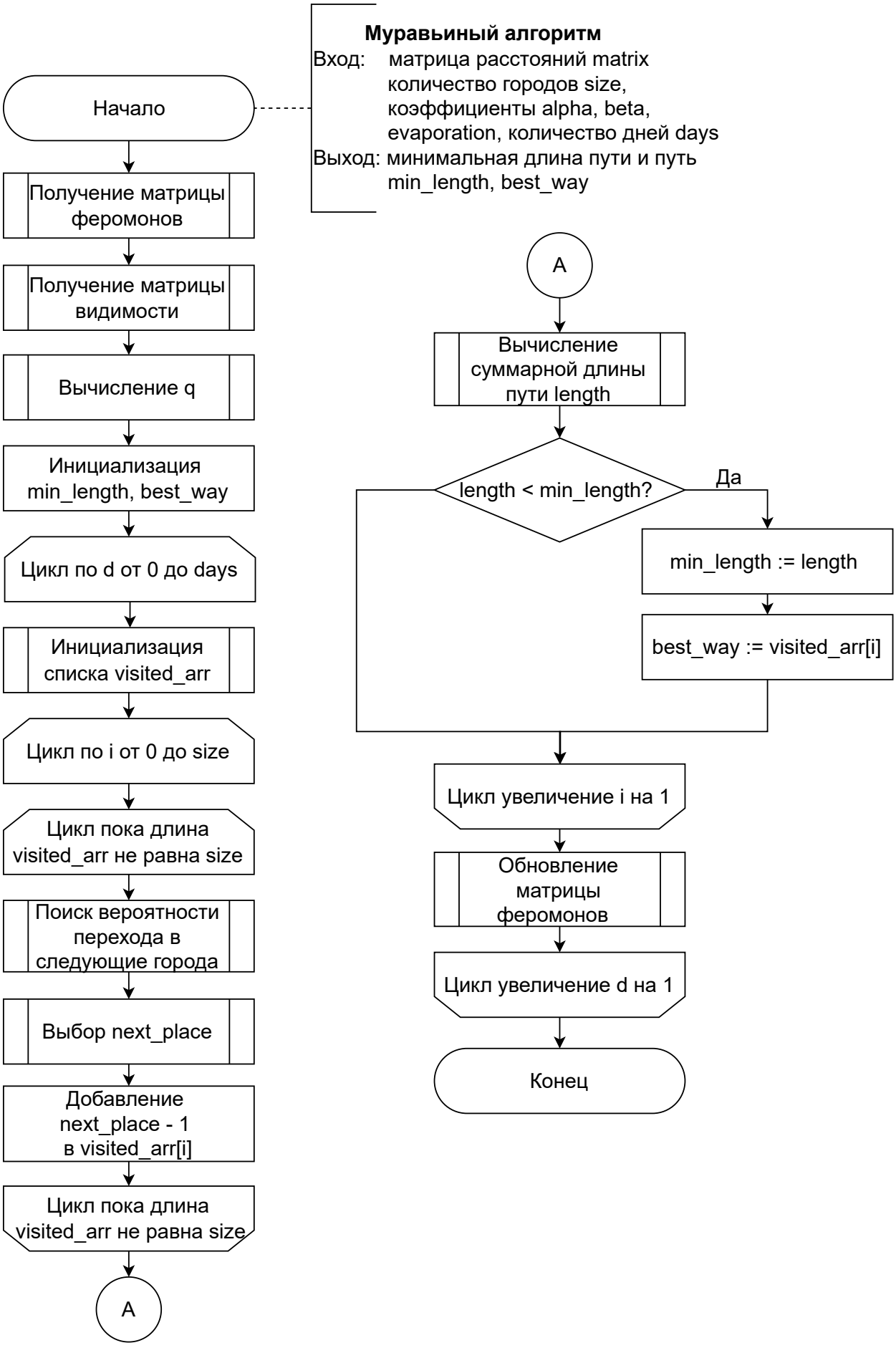

Рисунок 2.2 – Схема муравьиного алгоритма

Вывод

В данном разделе на основе теоретических данных были построены схемы требуемых алгоритмов.

3 Технологическая часть

3.1 Требования к программному обеспечению

В программе должна присутствовать возможность:

1) подавать на вход файл, содержащий матрицу смежности (расстояний), для которой можно решать задачу коммивояжера (поиск незамкнутого пути) алгоритмом полного перебора или муравьиным алгоритмом на выбор;

2) проводить параметризацию муравьиного алгоритма для матрицы смежности, считанной из файла;

3) замерять процессорное время выполнения алгоритмов.

3.2 Выбор языка программирования

Для реализации программного обеспечения был выбран язык *Python* в силу наличия модуля *numpy*, предоставляющего большое количество функций работы с массивами и матрицами.

3.3 Выбор библиотеки и способа для замера времени

Для замера процессорного времени выполнения реализаций агоритмов была выбрана функция *process_time_ns*() из библиотеки *time*.

3.4 Реализации алгоритмов

В листингах 3.1, 3.2 приведена реализация алгоритма полного перебора и муравьиного алгоритма соответственно.

Листинг 3.1 – Реализация алгоритма полного перебора

```
def calc_length(matrix, size, way):
    length = 0

    for i in range(size):
        beg_city = way[i]
        end_city = way[i + 1]

        length += matrix[beg_city][end_city]

    return length

def full_combinations_alg(matrix, size):
    cities = np.arange(size)
    cities_combs = []

    for combination in it.permutations(cities):
        cities_combs.append(list(combination))

    best_way = []
    min_length = float("inf")

    for i in range(len(cities_combs)):
        length = calc_length(matrix, size - 1, cities_combs[i])

        if length < min_length:
            min_length = length
            best_way = cities_combs[i]

    return min_length, best_way
```

Листинг 3.2 – Реализация муравьиного алгорима

```
def calc_q(matrix, size):
    q = 0
    count = 0

    for i in range(size):
        for j in range(size):
            if i != j:
                q += matrix[i][j]
                count += 1

    return q / count

def get_pherom_matr(size):
    pherom_matr = [[1 for i in range(size)]
                        for j in range(size)]

    return pherom_matr

def get_visib_matr(matrix, size):
    visib_matr = [[(1.0 / matrix[i][j] if (i != j) else 0)
                        for j in range(size)]
                            for i in range(size)]

    return visib_matr

def get_visited_places(route, ants):
    visited_arr = [[] for i in range(ants)]

    for i in range(ants):
        visited_arr[i].append(route[i])

    return visited_arr

def update_pherom_matr(matrix, size, visited_arr, pherom_matr, q,
    evaporation):
    ants = size

    for i in range(size):
        for j in range(size):
            delta = 0
```

```
40
41              for ant in range(ants):
42                  length = calc_length(matrix, size, visited_arr[ant
                        ])
43                  delta += q / length
44
45              pherom_matr[i][j] *= (1 - evaporation)
46              pherom_matr[i][j] += delta
47
48              if pherom_matr[i][j] < MIN_PHEROMONE:
49                  pherom_matr[i][j] = MIN_PHEROMONE
50
51      return pherom_matr
52
53  def choose_next_place(pk):
54      size = len(pk)
55      numb = 0
56      i = 0
57
58      probability = random()
59
60      while numb < probability and i < size:
61          numb += pk[i]
62          i += 1
63
64      return i
65
66  def search_probability(pherom_matr, visib_matr, visited_arr, size,
        ant, alpha, beta):
67      pk = [0] * size
68
69      for i in range(size):
70          if i not in visited_arr[ant]:
71              ant_i = visited_arr[ant][-1]
72
73              pk[i] = pow(pherom_matr[ant_i][i], alpha) * \
74                      pow(visib_matr[ant_i][i], beta)
75          else:
76              pk[i] = 0
77
78      pk_sum = sum(pk)
```

```
79
80      for i in range(size):
81          pk[i] /= pk_sum
82
83      return pk
84
85  def ant_alg(matrix, size, alpha, beta, evaporation, days):
86      pherom_matr = get_pherom_matr(size)
87      visib_matr  = get_visib_matr(matrix, size)
88
89      q = calc_q(matrix, size)
90
91      best_way = []
92      min_length = float("inf")
93
94      for j in range(days):
95          visited_arr = get_visited_places(np.arange(size), size)
96
97          for i in range(size):
98              while len(visited_arr[i]) != size:
99                  pk = search_probability(pherom_matr, visib_matr,
                        visited_arr, size, i, alpha, beta)
100                 next_place = choose_next_place(pk)
101
102                 visited_arr[i].append(next_place - 1)
103
104             length = calc_length(matrix, size - 1, visited_arr[i])
105
106             if length < min_length:
107                 min_length = length
108                 best_way = visited_arr[i]
109
110         pherom_matr = update_pherom_matr(matrix, size - 1,
                visited_arr, pherom_matr, q, evaporation)
111
112     return min_length, best_way
```

3.5 Тестирование алгоритмов

В таблице 3.1 приведены негативные функциональные тесты, в 3.2 — позитивные функциональные тесты для функций, реализующих алгоритм полного перебора или муравьиный алгоритм. Все тесты пройдены успешно.

Таблица 3.1 – Негативные функциональные тесты

Класс эквивалентности	Ожидаемый результат
Номер команды не число	Введено не целое число от 0 до 4
Номер команды < 0 или > 4	Введено не целое число от 0 до 4
Файл не существует	Невозможно прочитать данные файла
Файл содержит не числа	Невозможно прочитать данные файла
Коэффициент α не число	Введены неверные коэффициенты
Коэффициент $\alpha <= 0$	Введены неверные коэффициенты
Коэффициент β не число	Введены неверные коэффициенты
Коэффициент $\beta <= 0$	Введены неверные коэффициенты
Коэффициент $evaporation$ не число	Введены неверные коэффициенты
Коэффициент $evaporation <= 0$	Введены неверные коэффициенты
Коэффициент $days$ не число	Введены неверные коэффициенты
Коэффициент $days <= 0$	Введены неверные коэффициенты

Таблица 3.2 – Позитивные функциональные тесты

Матрица расстоян	Ожидаемый результат
$\begin{pmatrix} 0 & 37 \\ 37 & 0 \end{pmatrix}$	Минимальная сумма пути: 37 Минимальный путь: [0, 1]
$\begin{pmatrix} 0 & 59 & 13 & 14 & 85 \\ 59 & 0 & 98 & 25 & 80 \\ 13 & 98 & 0 & 8 & 92 \\ 14 & 25 & 8 & 0 & 99 \\ 85 & 80 & 92 & 99 & 0 \end{pmatrix}$	Минимальная сумма пути: 126 Минимальный путь: [0, 2, 3, 1, 4]

Вывод

В данном разделе были реализованы алгоритм полного перебора, муравьиный алгоритм, а также проведено функциональное тестирование данных алгоритмов.

4 Экспериментальная часть

4.1 Технические характеристики

Ниже приведены технические характеристики устройства, на котором было проведено тестирование программного обеспечения:

1) операционная система Windows-10, 64-bit;
2) оперативная память 8 ГБ;
3) процессор Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 2304 МГц, ядер 2, логических процессоров 4.

4.2 Замеры времени

В таблице 4.1 приведены результаты замеров в миллисекундах времени работы алгоритмов полного перебора и муравьиного.

Таблица 4.1 – Замеры времени выполнения алгоритмов на произвольных массивах разной длины

Кол-во узлов	Полный перебор	Муравьиный
6	17.600	15.625
7	31.250	62.500
8	390.625	109.375
9	3062.500	140.750
10	36625.000	171.875

Зависимость времени работы алгоритмов полного перебора и муравьиного представлена на рисунке 4.1. Однако в силу факториальной сложности алгоритма полного перебора удается рассмотреть только небольшую дельту размерностей матрицы, так как далее график становится нечитаемым.

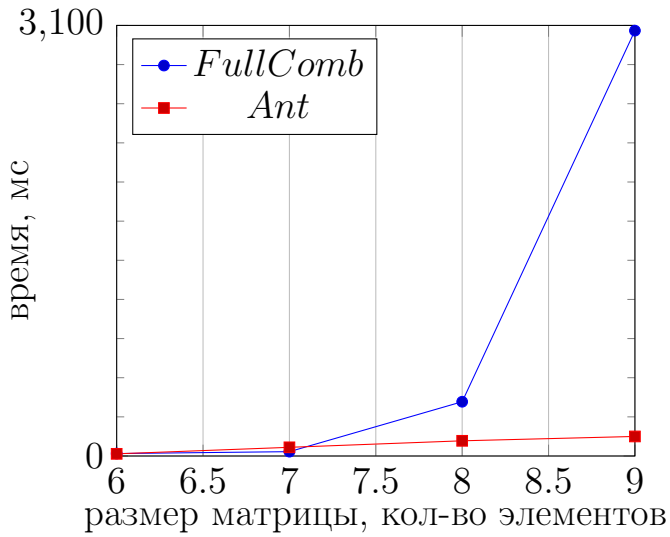

Рисунок 4.1 – Зависимость времени работы алгоритмов последовательной и конвейерной обработки для разного количества заявок

4.3 Автоматическая парамертризация

Автоматическая параметризация была проведена на двух классах данных. Для проведение эксперимета были взяты матрицы размером 10x10. Муравьиный алгоритм был запущен для всех значений $\alpha, \rho \in (0, 1)$ с шагом 0.1. В качестве эталонного значения был взят результат работы алгоритма полного перебора.

4.3.1 Первый класс данных

Матрица расстояний первого класса данных:

$$M_1 = \begin{pmatrix} 0 & 2 & 2 & 1 & 2 & 3 & 2 & 2 & 1 & 1 \\ 2 & 0 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ 2 & 2 & 0 & 1 & 2 & 2 & 3 & 3 & 1 & 1 \\ 1 & 3 & 1 & 0 & 3 & 3 & 3 & 2 & 2 & 2 \\ 2 & 2 & 2 & 3 & 0 & 3 & 1 & 1 & 1 & 3 \\ 3 & 3 & 2 & 3 & 3 & 0 & 1 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 & 1 & 1 & 0 & 2 & 1 & 1 \\ 2 & 3 & 3 & 2 & 1 & 2 & 2 & 0 & 1 & 3 \\ 1 & 2 & 1 & 2 & 1 & 2 & 1 & 1 & 0 & 3 \\ 1 & 1 & 1 & 2 & 3 & 2 & 1 & 3 & 3 & 0 \end{pmatrix} \tag{4.1}$$

В таблице 4.2 представлены данные, полученные в результате параметризации для первого класса данных.

Таблица 4.2 – Параметры для первого класса данных

α	β	ρ	Дней	Результат	Ошибка
0.1	0.9	0.1	50	9	1
0.1	0.9	0.1	100	9	2
0.1	0.9	0.1	200	9	1
0.1	0.9	0.2	50	9	2
0.1	0.9	0.2	100	9	2
0.1	0.9	0.2	200	9	1
0.1	0.9	0.3	50	9	1
0.1	0.9	0.3	100	9	1
0.1	0.9	0.3	200	9	1
0.1	0.9	0.4	50	9	2
0.1	0.9	0.4	100	9	0
0.1	0.9	0.4	200	9	1
0.1	0.9	0.5	50	9	2
0.1	0.9	0.5	100	9	1
0.1	0.9	0.5	200	9	1
0.1	0.9	0.6	50	9	2
0.1	0.9	0.6	100	9	1

0.1	0.9	0.6	200	9	1
0.1	0.9	0.7	50	9	2
0.1	0.9	0.7	100	9	2
0.1	0.9	0.7	200	9	1
0.1	0.9	0.8	50	9	2
0.1	0.9	0.8	100	9	1
0.1	0.9	0.8	200	9	1
0.2	0.8	0.1	50	9	2
0.2	0.8	0.1	100	9	2
0.2	0.8	0.1	200	9	1
0.2	0.8	0.2	50	9	3
0.2	0.8	0.2	100	9	2
0.2	0.8	0.2	200	9	1
0.2	0.8	0.3	50	9	2
0.2	0.8	0.3	100	9	2
0.2	0.8	0.3	200	9	1
0.2	0.8	0.4	50	9	1
0.2	0.8	0.4	100	9	1
0.2	0.8	0.4	200	9	2
0.2	0.8	0.5	50	9	2
0.2	0.8	0.5	100	9	0
0.2	0.8	0.5	200	9	1
0.2	0.8	0.6	50	9	0
0.2	0.8	0.6	100	9	1
0.2	0.8	0.6	200	9	0
0.2	0.8	0.7	50	9	1
0.2	0.8	0.7	100	9	2
0.2	0.8	0.7	200	9	1
0.2	0.8	0.8	50	9	1
0.2	0.8	0.8	100	9	1
0.2	0.8	0.8	200	9	1
0.3	0.7	0.1	50	9	2
0.3	0.7	0.1	100	9	2
0.3	0.7	0.1	200	9	2

0.3	0.7	0.2	50	9	2
0.3	0.7	0.2	100	9	2
0.3	0.7	0.2	200	9	1
0.3	0.7	0.3	50	9	2
0.3	0.7	0.3	100	9	1
0.3	0.7	0.3	200	9	1
0.3	0.7	0.4	50	9	2
0.3	0.7	0.4	100	9	2
0.3	0.7	0.4	200	9	1
0.3	0.7	0.5	50	9	3
0.3	0.7	0.5	100	9	1
0.3	0.7	0.5	200	9	2
0.3	0.7	0.6	50	9	2
0.3	0.7	0.6	100	9	2
0.3	0.7	0.6	200	9	1
0.3	0.7	0.7	50	9	2
0.3	0.7	0.7	100	9	2
0.3	0.7	0.7	200	9	1
0.3	0.7	0.8	50	9	2
0.3	0.7	0.8	100	9	2
0.3	0.7	0.8	200	9	1
0.4	0.6	0.1	50	9	2
0.4	0.6	0.1	100	9	2
0.4	0.6	0.1	200	9	2
0.4	0.6	0.2	50	9	2
0.4	0.6	0.2	100	9	2
0.4	0.6	0.2	200	9	2
0.4	0.6	0.3	50	9	2
0.4	0.6	0.3	100	9	2
0.4	0.6	0.3	200	9	1
0.4	0.6	0.4	50	9	2
0.4	0.6	0.4	100	9	2
0.4	0.6	0.4	200	9	1

0.4	0.6	0.5	50	9	3
0.4	0.6	0.5	100	9	2
0.4	0.6	0.5	200	9	1
0.4	0.6	0.6	50	9	3
0.4	0.6	0.6	100	9	1
0.4	0.6	0.6	200	9	1
0.4	0.6	0.7	50	9	3
0.4	0.6	0.7	100	9	2
0.4	0.6	0.7	200	9	2
0.4	0.6	0.8	50	9	2
0.4	0.6	0.8	100	9	2
0.4	0.6	0.8	200	9	1
0.5	0.5	0.1	50	9	2
0.5	0.5	0.1	100	9	2
0.5	0.5	0.1	200	9	2
0.5	0.5	0.2	50	9	2
0.5	0.5	0.2	100	9	2
0.5	0.5	0.2	200	9	2
0.5	0.5	0.3	50	9	2
0.5	0.5	0.3	100	9	2
0.5	0.5	0.3	200	9	2
0.5	0.5	0.4	50	9	3
0.5	0.5	0.4	100	9	2
0.5	0.5	0.4	200	9	2
0.5	0.5	0.5	50	9	2
0.5	0.5	0.5	100	9	2
0.5	0.5	0.5	200	9	2
0.5	0.5	0.6	50	9	1
0.5	0.5	0.6	100	9	2
0.5	0.5	0.6	200	9	2
0.5	0.5	0.7	50	9	2
0.5	0.5	0.7	100	9	1
0.5	0.5	0.7	200	9	1

0.5	0.5	0.8	50	9	2
0.5	0.5	0.8	100	9	2
0.5	0.5	0.8	200	9	1
0.6	0.4	0.1	50	9	3
0.6	0.4	0.1	100	9	2
0.6	0.4	0.1	200	9	2
0.6	0.4	0.2	50	9	2
0.6	0.4	0.2	100	9	2
0.6	0.4	0.2	200	9	2
0.6	0.4	0.3	50	9	3
0.6	0.4	0.3	100	9	2
0.6	0.4	0.3	200	9	2
0.6	0.4	0.4	50	9	2
0.6	0.4	0.4	100	9	2
0.6	0.4	0.4	200	9	2
0.6	0.4	0.5	50	9	3
0.6	0.4	0.5	100	9	2
0.6	0.4	0.5	200	9	2
0.6	0.4	0.6	50	9	2
0.6	0.4	0.6	100	9	2
0.6	0.4	0.6	200	9	2
0.6	0.4	0.7	50	9	3
0.6	0.4	0.7	100	9	2
0.6	0.4	0.7	200	9	2
0.6	0.4	0.8	50	9	2
0.6	0.4	0.8	100	9	3
0.6	0.4	0.8	200	9	2
0.7	0.3	0.1	50	9	2
0.7	0.3	0.1	100	9	2
0.7	0.3	0.1	200	9	3
0.7	0.3	0.2	50	9	3
0.7	0.3	0.2	100	9	2
0.7	0.3	0.2	200	9	1

0.7	0.3	0.3	50	9	3
0.7	0.3	0.3	100	9	3
0.7	0.3	0.3	200	9	3
0.7	0.3	0.4	50	9	3
0.7	0.3	0.4	100	9	2
0.7	0.3	0.4	200	9	2
0.7	0.3	0.5	50	9	3
0.7	0.3	0.5	100	9	3
0.7	0.3	0.5	200	9	2
0.7	0.3	0.6	50	9	3
0.7	0.3	0.6	100	9	2
0.7	0.3	0.6	200	9	2
0.7	0.3	0.7	50	9	1
0.7	0.3	0.7	100	9	2
0.7	0.3	0.7	200	9	1
0.7	0.3	0.8	50	9	2
0.7	0.3	0.8	100	9	2
0.7	0.3	0.8	200	9	3
0.8	0.2	0.1	50	9	3
0.8	0.2	0.1	100	9	2
0.8	0.2	0.1	200	9	2
0.8	0.2	0.2	50	9	3
0.8	0.2	0.2	100	9	3
0.8	0.2	0.2	200	9	2
0.8	0.2	0.3	50	9	2
0.8	0.2	0.3	100	9	3
0.8	0.2	0.3	200	9	2
0.8	0.2	0.4	50	9	2
0.8	0.2	0.4	100	9	2
0.8	0.2	0.4	200	9	2
0.8	0.2	0.5	50	9	3
0.8	0.2	0.5	100	9	2
0.8	0.2	0.5	200	9	2

0.8	0.2	0.6	50	9	2
0.8	0.2	0.6	100	9	2
0.8	0.2	0.6	200	9	3
0.8	0.2	0.7	50	9	3
0.8	0.2	0.7	100	9	2
0.8	0.2	0.7	200	9	2
0.8	0.2	0.8	50	9	3
0.8	0.2	0.8	100	9	3
0.8	0.2	0.8	200	9	3
0.9	0.1	0.1	50	9	2
0.9	0.1	0.1	100	9	2
0.9	0.1	0.1	200	9	2
0.9	0.1	0.2	50	9	2
0.9	0.1	0.2	100	9	2
0.9	0.1	0.2	200	9	2
0.9	0.1	0.3	50	9	2
0.9	0.1	0.3	100	9	3
0.9	0.1	0.3	200	9	1
0.9	0.1	0.4	50	9	3
0.9	0.1	0.4	100	9	1
0.9	0.1	0.4	200	9	2
0.9	0.1	0.5	50	9	3
0.9	0.1	0.5	100	9	2
0.9	0.1	0.5	200	9	2
0.9	0.1	0.6	50	9	3
0.9	0.1	0.6	100	9	3
0.9	0.1	0.6	200	9	1
0.9	0.1	0.7	50	9	2
0.9	0.1	0.7	100	9	3
0.9	0.1	0.7	200	9	3
0.9	0.1	0.8	50	9	3
0.9	0.1	0.8	100	9	3
0.9	0.1	0.8	200	9	2

4.3.2 Второй класс данных

Матрица расстояний второго класса данных:

$$M_2 = \begin{pmatrix} 0 & 29 & 44 & 73 & 80 & 24 & 71 & 56 & 40 & 70 \\ 29 & 0 & 2 & 42 & 41 & 6 & 26 & 96 & 5 & 43 \\ 44 & 2 & 0 & 55 & 80 & 56 & 99 & 9 & 61 & 34 \\ 73 & 42 & 55 & 0 & 27 & 18 & 45 & 77 & 92 & 11 \\ 80 & 41 & 80 & 27 & 0 & 33 & 19 & 55 & 4 & 96 \\ 24 & 6 & 56 & 18 & 33 & 0 & 76 & 74 & 53 & 14 \\ 71 & 26 & 99 & 45 & 19 & 76 & 0 & 44 & 52 & 55 \\ 56 & 96 & 9 & 77 & 55 & 74 & 44 & 0 & 82 & 57 \\ 40 & 5 & 61 & 92 & 4 & 53 & 52 & 82 & 0 & 78 \\ 70 & 43 & 34 & 11 & 96 & 14 & 55 & 57 & 78 & 0 \end{pmatrix} \tag{4.2}$$

В таблице 4.3 представлены данные, полученные в результате параметризации для второго класса данных.

Таблица 4.3 – Параметры для второго класса данных

α	β	ρ	Дней	Результат	Ошибка
0.1	0.9	0.1	50	133	7
0.1	0.9	0.1	100	133	0
0.1	0.9	0.1	200	133	0
0.1	0.9	0.2	50	133	20
0.1	0.9	0.2	100	133	16
0.1	0.9	0.2	200	133	7
0.1	0.9	0.3	50	133	16
0.1	0.9	0.3	100	133	0
0.1	0.9	0.3	200	133	0
0.1	0.9	0.4	50	133	28
0.1	0.9	0.4	100	133	7
0.1	0.9	0.4	200	133	7

0.1	0.9	0.5	50	133	0
0.1	0.9	0.5	100	133	11
0.1	0.9	0.5	200	133	7
0.1	0.9	0.6	50	133	0
0.1	0.9	0.6	100	133	11
0.1	0.9	0.6	200	133	0
0.1	0.9	0.7	50	133	24
0.1	0.9	0.7	100	133	7
0.1	0.9	0.7	200	133	0
0.1	0.9	0.8	50	133	7
0.1	0.9	0.8	100	133	0
0.1	0.9	0.8	200	133	11
0.2	0.8	0.1	50	133	0
0.2	0.8	0.1	100	133	15
0.2	0.8	0.1	200	133	7
0.2	0.8	0.2	50	133	25
0.2	0.8	0.2	100	133	14
0.2	0.8	0.2	200	133	14
0.2	0.8	0.3	50	133	28
0.2	0.8	0.3	100	133	0
0.2	0.8	0.3	200	133	0
0.2	0.8	0.4	50	133	0
0.2	0.8	0.4	100	133	7
0.2	0.8	0.4	200	133	11
0.2	0.8	0.5	50	133	26
0.2	0.8	0.5	100	133	0
0.2	0.8	0.5	200	133	0
0.2	0.8	0.6	50	133	20
0.2	0.8	0.6	100	133	0
0.2	0.8	0.6	200	133	0
0.2	0.8	0.7	50	133	16
0.2	0.8	0.7	100	133	0
0.2	0.8	0.7	200	133	0

0.2	0.8	0.8	50	133	23
0.2	0.8	0.8	100	133	15
0.2	0.8	0.8	200	133	11
0.3	0.7	0.1	50	133	40
0.3	0.7	0.1	100	133	34
0.3	0.7	0.1	200	133	0
0.3	0.7	0.2	50	133	38
0.3	0.7	0.2	100	133	16
0.3	0.7	0.2	200	133	16
0.3	0.7	0.3	50	133	20
0.3	0.7	0.3	100	133	20
0.3	0.7	0.3	200	133	16
0.3	0.7	0.4	50	133	25
0.3	0.7	0.4	100	133	16
0.3	0.7	0.4	200	133	11
0.3	0.7	0.5	50	133	0
0.3	0.7	0.5	100	133	18
0.3	0.7	0.5	200	133	7
0.3	0.7	0.6	50	133	27
0.3	0.7	0.6	100	133	32
0.3	0.7	0.6	200	133	15
0.3	0.7	0.7	50	133	42
0.3	0.7	0.7	100	133	11
0.3	0.7	0.7	200	133	7
0.3	0.7	0.8	50	133	0
0.3	0.7	0.8	100	133	16
0.3	0.7	0.8	200	133	7
0.4	0.6	0.1	50	133	51
0.4	0.6	0.1	100	133	30
0.4	0.6	0.1	200	133	0
0.4	0.6	0.2	50	133	0
0.4	0.6	0.2	100	133	11
0.4	0.6	0.2	200	133	15

0.4	0.6	0.3	50	133	18
0.4	0.6	0.3	100	133	22
0.4	0.6	0.3	200	133	7
0.4	0.6	0.4	50	133	16
0.4	0.6	0.4	100	133	7
0.4	0.6	0.4	200	133	7
0.4	0.6	0.5	50	133	38
0.4	0.6	0.5	100	133	22
0.4	0.6	0.5	200	133	23
0.4	0.6	0.6	50	133	0
0.4	0.6	0.6	100	133	34
0.4	0.6	0.6	200	133	7
0.4	0.6	0.7	50	133	37
0.4	0.6	0.7	100	133	14
0.4	0.6	0.7	200	133	27
0.4	0.6	0.8	50	133	39
0.4	0.6	0.8	100	133	0
0.4	0.6	0.8	200	133	28
0.5	0.5	0.1	50	133	62
0.5	0.5	0.1	100	133	39
0.5	0.5	0.1	200	133	34
0.5	0.5	0.2	50	133	28
0.5	0.5	0.2	100	133	36
0.5	0.5	0.2	200	133	27
0.5	0.5	0.3	50	133	0
0.5	0.5	0.3	100	133	34
0.5	0.5	0.3	200	133	20
0.5	0.5	0.4	50	133	48
0.5	0.5	0.4	100	133	27
0.5	0.5	0.4	200	133	24
0.5	0.5	0.5	50	133	30
0.5	0.5	0.5	100	133	44
0.5	0.5	0.5	200	133	11

0.5	0.5	0.6	50	133	38
0.5	0.5	0.6	100	133	29
0.5	0.5	0.6	200	133	24
0.5	0.5	0.7	50	133	37
0.5	0.5	0.7	100	133	44
0.5	0.5	0.7	200	133	16
0.5	0.5	0.8	50	133	15
0.5	0.5	0.8	100	133	20
0.5	0.5	0.8	200	133	15
0.6	0.4	0.1	50	133	63
0.6	0.4	0.1	100	133	36
0.6	0.4	0.1	200	133	22
0.6	0.4	0.2	50	133	37
0.6	0.4	0.2	100	133	27
0.6	0.4	0.2	200	133	61
0.6	0.4	0.3	50	133	61
0.6	0.4	0.3	100	133	15
0.6	0.4	0.3	200	133	34
0.6	0.4	0.4	50	133	60
0.6	0.4	0.4	100	133	45
0.6	0.4	0.4	200	133	24
0.6	0.4	0.5	50	133	70
0.6	0.4	0.5	100	133	48
0.6	0.4	0.5	200	133	42
0.6	0.4	0.6	50	133	37
0.6	0.4	0.6	100	133	46
0.6	0.4	0.6	200	133	17
0.6	0.4	0.7	50	133	27
0.6	0.4	0.7	100	133	0
0.6	0.4	0.7	200	133	24
0.6	0.4	0.8	50	133	20
0.6	0.4	0.8	100	133	66
0.6	0.4	0.8	200	133	34

0.7	0.3	0.1	50	133	64
0.7	0.3	0.1	100	133	68
0.7	0.3	0.1	200	133	30
0.7	0.3	0.2	50	133	31
0.7	0.3	0.2	100	133	27
0.7	0.3	0.2	200	133	15
0.7	0.3	0.3	50	133	55
0.7	0.3	0.3	100	133	53
0.7	0.3	0.3	200	133	24
0.7	0.3	0.4	50	133	51
0.7	0.3	0.4	100	133	27
0.7	0.3	0.4	200	133	34
0.7	0.3	0.5	50	133	46
0.7	0.3	0.5	100	133	23
0.7	0.3	0.5	200	133	35
0.7	0.3	0.6	50	133	83
0.7	0.3	0.6	100	133	45
0.7	0.3	0.6	200	133	46
0.7	0.3	0.7	50	133	40
0.7	0.3	0.7	100	133	48
0.7	0.3	0.7	200	133	48
0.7	0.3	0.8	50	133	61
0.7	0.3	0.8	100	133	67
0.7	0.3	0.8	200	133	7
0.8	0.2	0.1	50	133	64
0.8	0.2	0.1	100	133	39
0.8	0.2	0.1	200	133	45
0.8	0.2	0.2	50	133	56
0.8	0.2	0.2	100	133	78
0.8	0.2	0.2	200	133	7
0.8	0.2	0.3	50	133	29
0.8	0.2	0.3	100	133	39
0.8	0.2	0.3	200	133	34

0.8	0.2	0.4	50	133	66
0.8	0.2	0.4	100	133	68
0.8	0.2	0.4	200	133	51
0.8	0.2	0.5	50	133	51
0.8	0.2	0.5	100	133	57
0.8	0.2	0.5	200	133	20
0.8	0.2	0.6	50	133	87
0.8	0.2	0.6	100	133	67
0.8	0.2	0.6	200	133	51
0.8	0.2	0.7	50	133	66
0.8	0.2	0.7	100	133	52
0.8	0.2	0.7	200	133	60
0.8	0.2	0.8	50	133	61
0.8	0.2	0.8	100	133	71
0.8	0.2	0.8	200	133	44
0.9	0.1	0.1	50	133	90
0.9	0.1	0.1	100	133	66
0.9	0.1	0.1	200	133	38
0.9	0.1	0.2	50	133	110
0.9	0.1	0.2	100	133	59
0.9	0.1	0.2	200	133	34
0.9	0.1	0.3	50	133	80
0.9	0.1	0.3	100	133	59
0.9	0.1	0.3	200	133	63
0.9	0.1	0.4	50	133	87
0.9	0.1	0.4	100	133	85
0.9	0.1	0.4	200	133	44
0.9	0.1	0.5	50	133	60
0.9	0.1	0.5	100	133	44
0.9	0.1	0.5	200	133	56
0.9	0.1	0.6	50	133	89
0.9	0.1	0.6	100	133	70
0.9	0.1	0.6	200	133	16

0.9	0.1	0.7	50	133	53
0.9	0.1	0.7	100	133	27
0.9	0.1	0.7	200	133	36
0.9	0.1	0.8	50	133	69
0.9	0.1	0.8	100	133	50
0.9	0.1	0.8	200	133	72

Вывод

В данном разделе был проведен эксперимент по измерению времени работы алгоритма полного перебора и муравьиного алгоритма. Согласно полученным при проведении эксперимента данным, муравьиный алгоритм работает быстрее при всех рассматренных условиях. А также была проведена параметризация на двух классах данных, что позволило выяснить, наиболее подходящий параметры для классов данных.

Для первого класса данных:

- $\alpha = 0.1, \beta = 0.9, \rho = 0.3$;
- $\alpha = 0.1, \beta = 0.9, \rho = 0.4$;
- $\alpha = 0.2, \beta = 0.8, \rho = 0.5$;
- $\alpha = 0.2, \beta = 0.8, \rho = 0.6$;
- $\alpha = 0.2, \beta = 0.8, \rho = 0.8$.

Для второго класса данных:

- $\alpha = 0.1, \beta = 0.9, \rho = 0.1$;
- $\alpha = 0.1, \beta = 0.9, \rho = 0.3$;
- $\alpha = 0.1, \beta = 0.9, \rho = 0.6$;
- $\alpha = 0.2, \beta = 0.8, \rho = 0.7$.

Заключение

Цель данной лабораторной работы достигнута: изучен муравьиный алгоритма на примере задачи коммивояжера.

Все поставленные **задачи** выполнены:

1) исследована задача коммивояжера;
2) изучены алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжера;
3) проведена параметризация муравьиного алгоритма на нескольких классах данных;
4) составлены схемы используемых алгоритмов;
5) реализованы алгоритм полного перебора и муравьиный алгоритм;
6) проведен сравнительный анализ времени работы данных алгоритмов;
7) описаны и обоснованы полученные результаты в отчете.

На основании проведенных экспериментов было определено, что муравьиный алгоритм работает быстрее алгоритма полного перебора. А также уставновлено, что параметры наилучшей работы муравьиного алгоритма: $\alpha = 0.1, \beta = 0.9, \rho = 0.3$.

Список использованных источников

[1] Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. – 2003. – №4.

[2] Курейчик В. М., Кажаров А. А. О некоторых модификациях муравьиного алгоритма // Известия ЮФУ. Технические науки. 2008. №4.