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»

Отчёт по лабораторной работе №6 по курсу «Анализ Алгоритмов»

Студент Паламарчук А.Н.

Группа ИУ7-53Б

Преподаватель Кормановский М.В.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является исследование алгоритма полного перебора и муравьиного алгоритма для решения задачи коммивояжёра. Для достижения поставленной цели необходимо выполнить следующие задачи:

— разработать алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжёра;

— разработать программное обеспечение, для решения задачи коммивояжёра при помощи указанных алгоритмов;

— выполнить параметризацию муравьиного алгоритма по трём его параметрам;

— провести сравнительный анализ алгоритмов.

Индивидуальный вариант: неориентированный граф, без элитных муравьёв, карта городов древнего мира, незамкнутый маршрут.

1 Аналитическая часть

1.1 Задача коммивояжёра

Задача коммивояжёра формулируется следующим образом: необходимо найти такой кратчайший путь по заданным n городам, чтобы каждый город посещался только один раз. Проблема моделируется при помощи взвешенного графа, вершины которого представляют города, а веса рёбер определяют расстояния [1].

1.2 Алгоритм полного перебора

Алгоритм полного перебора [2] осуществляет поиск в пространстве $N!$ решений посредством перебора всех вариантов маршрутов. Преимуществом данного алгоритма заключается в том, что он гарантированно находит лучшее решение(глобальный минимум). Недостатком алгоритма полного перебора является его временная сложность — $O(N!)$, поэтому данный алгоритм целесообразно использовать, когда N является малым.

1.3 Муравьиный алгоритм

Муравьиный алгоритм [3] представляет собой вероятностную жадную эвристику, где вероятности устанавливаются, исходя из информации о качестве решения, полученной из предыдущих решений. 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 помечает путь феромоном, и эта информация используется другими муравьями для выбора пути.

У муравья три компетенции:

— зрение — муравей может определить привлекательность ребра;

— память — запоминает каждый посещённый в текущий день город в кортеж;

— обоняние — муравей чует концентрацию феромона на ребре.

Видимость — величина, обратная расстоянию: $\eta_{ij} = 1/D_{ij}$, где D_{ij} — расстояние между городами i и j.

Вероятностно-пропорциональное правило, определяющее вероятность перехода муравья k из текущей вершины i в вершину j на t итерации рассчитывается по формуле (1.1):

$$P_{kij}(t) = \begin{cases} \frac{\eta_{ij}^{\alpha}(\tau_{ij}(t))^{\beta}}{\sum_{q=1}^{N}\eta_{ij}^{\alpha}(\tau_{iq}(t))^{\beta}}, \text{если вершина } j \text{ ещё не посещена муравьём } k, \\ 0, \text{иначе,} \end{cases} \tag{1.1}$$

где τ_{ij} — количество феромонов на ребре (i, j), α — коэффициент жадности алгоритма, β — коэффициент стадности алгоритма.

После завершения движения всех муравьев осуществляется пересчёт уровня феромона для каждого ребра по следующей формуле (1.2):

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{1.2}$$

где $p \in (0, 1)$ — коэффициент испарения феромона, $\Delta\tau_{ij}(t)$ вычисляется по формуле:

$$\Delta\tau_{ij} = \sum_{k=1}^{N} \Delta\tau_{ijk}(t), \tag{1.3}$$

$$\Delta\tau_{ijk}(t) = \begin{cases} 0, \text{если по ребру } i-j \text{ муравей } k \text{ в день } t \text{ не ходил,} \\ \frac{Q}{L_k}(t), \text{иначе} \end{cases} \tag{1.4}$$

где Q — дневная квота феромона муравья, величина соизмеримая длине лучшего маршрута, а L_k — длина маршрута муравья k.

2 Конструкторская часть

2.1 Описание используемых структур данных

При реализации алгоритмов будут использованы следующие структуры данных:

— матрица — массив массивов целочисленного типа;

— размерность матрицы — целочисленный тип.

2.2 Разработка алгоритмов

На рисунках 2.1 и 2.2 представлены алгоритм полного перебора и муравьиный алгоритм.

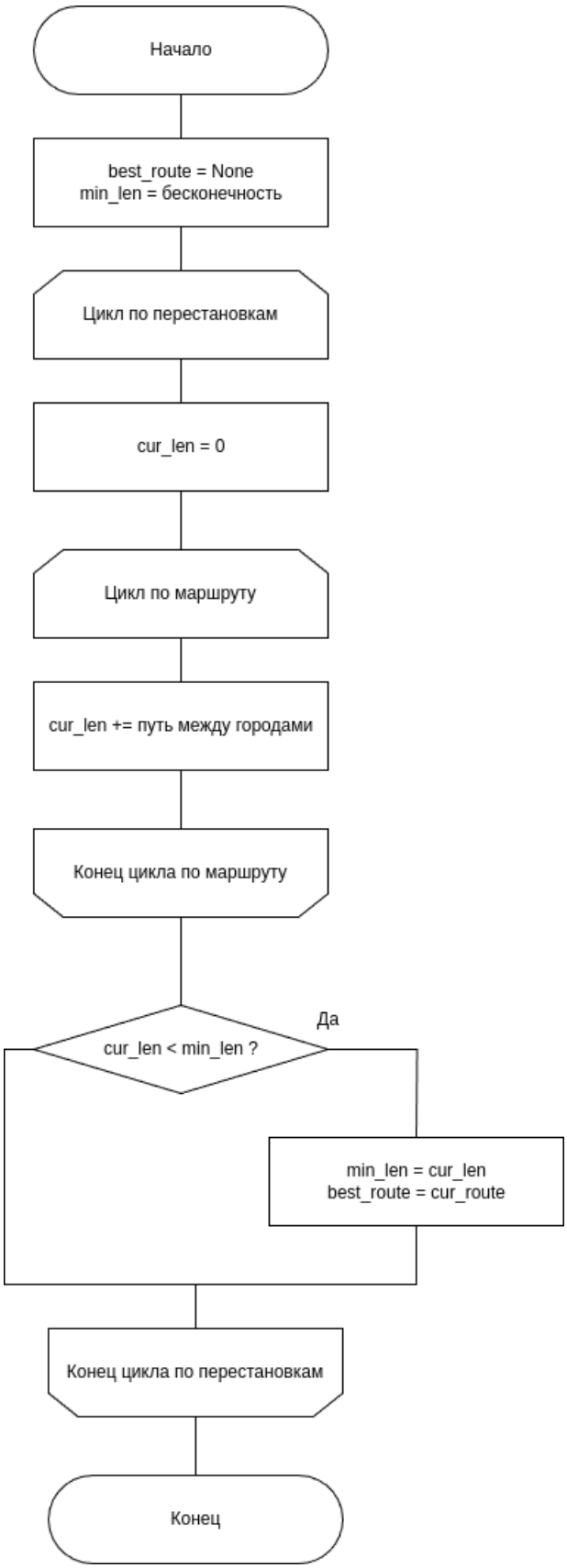

Рисунок 2.1 – Схема алгоритма полного перебора

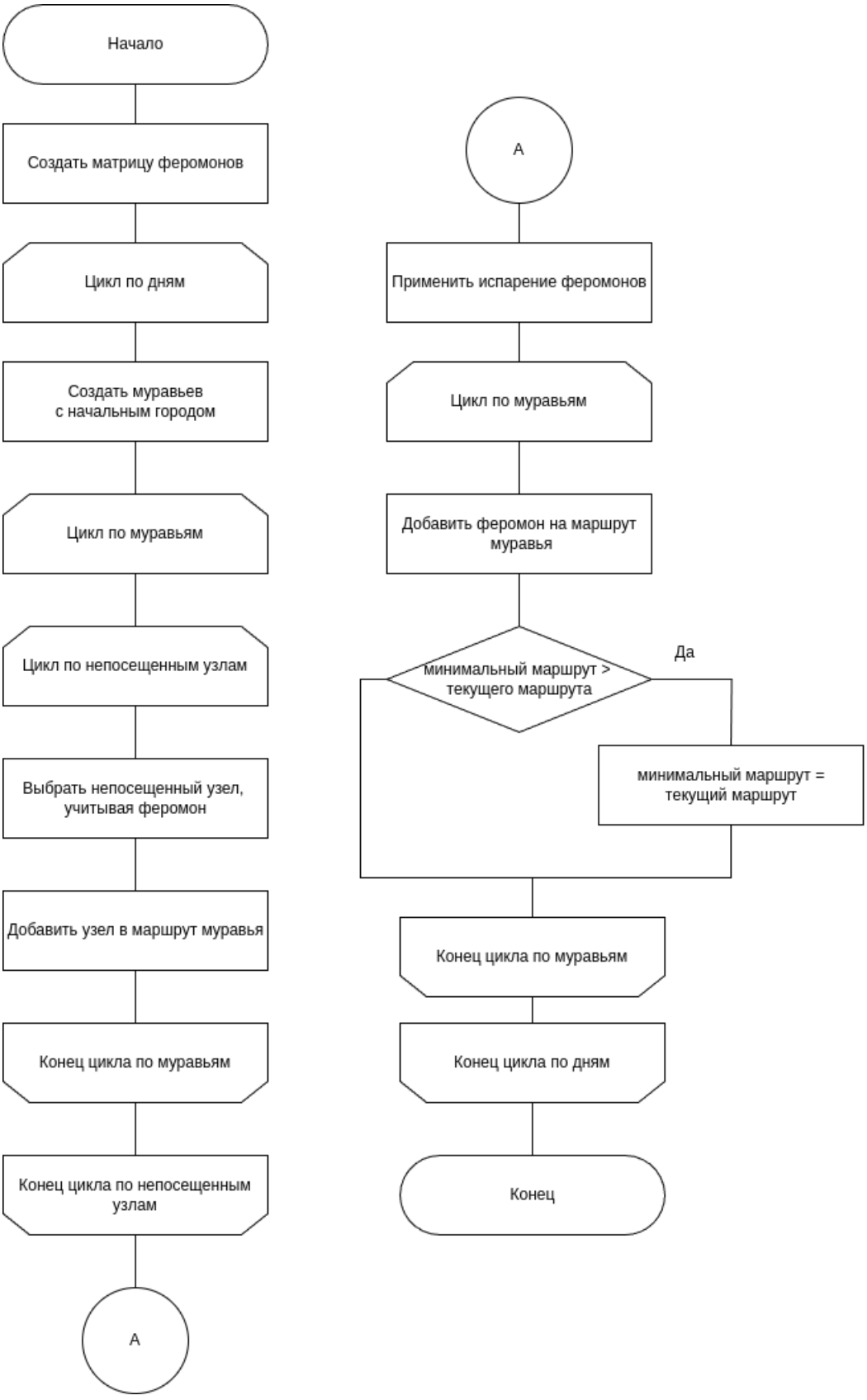

Рисунок 2.2 – Схема муравьиного алгоритма

3 Технологическая часть

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python*. Требуется измерить временные затраты и построить графики. Для построения графиков использовалась библиотека *mathplotlib*.

3.2 Реализация алгоритмов

В листингах 3.1- 3.2 представлены реализации алгоритма полного перебора и муравьиного алгоритма для решения задачи коммивояжёра.

Листинг 3.1 – Алгоритм полного перебора

```
def brute_force_voyager(matrix, size):
    cities = list(range(size))
    permutations = it.permutations(cities)

    min_len = float("inf")
    best_route = None
    for perm in permutations:
        cur_route = list(perm)
        cur_len = 0

        for i in range(len(cur_route) - 1):
            cur_len += matrix[cur_route[i]][cur_route[i + 1]]

        if cur_len < min_len:
            min_len = cur_len
            best_route = cur_route

    return min_len, best_route
```

Листинг 3.2 – Муравьиный алгоритм

```
1  def daily_quota_Q(matrix, size):
2      matrix = np.array(matrix)
3      total_sum = np.sum(matrix) - np.sum(np.diag(matrix))
4      count = size * (size - 1)
5      return total_sum / count
6
7  def calc_len_route(matrix, route):
8      length = sum(matrix[route[i], route[i + 1]] for i in
           range(len(route) - 1))
9      return length
10
11 def update_pheromones(matrix, size, visited, pheromones,
       daily_quota, k_evaporation):
12     for i in range(size):
13         for j in range(size):
14             delta_pheromones = 0
15             for ant in range(size):
16                 length = calc_len_route(matrix, visited[ant])
17                 delta_pheromones += daily_quota / length
18
19             pheromones[i][j] = pheromones[i][j] * (1 -
                   k_evaporation) + delta_pheromones
20             pheromones[i][j] = max(pheromones[i][j], MIN_PHEROMONE)
21
22     return pheromones
23
24 def get_matrix_pheromones(size):
25     return [[INIT_PHEROMONE] * size for _ in range(size)]
26
27 def get_matrix_visibility(matrix, size):
28     visibility_matrix = []
29     for i in range(size):
30         row = []
31         for j in range(size):
32             if i != j:
33                 row.append(1.0 / matrix[i][j])
34             else:
35                 row.append(0)
36         visibility_matrix.append(row)
37
```

```
38      return visibility_matrix
39
40  def get_matrix_visited(route, ants):
41      visited = [[] for _ in range(ants)]
42      for ant in range(ants):
43          visited[ant].append(int(route[ant]))
44
45      return visited
46
47  def choose_next_city(pheromones, ant, alpha, beta, visibility,
        visited, cities):
48      probabilities = [0] * cities
49      for city in range(cities):
50          if city not in visited[ant]:
51              ant_city = visited[ant][-1]
52              probabilities[city] = (pheromones[ant_city][city] **
                    alpha) * (visibility[ant_city][city] ** beta)
53          else:
54              probabilities[city] = 0
55
56      sum_probabilities = sum(probabilities)
57      if sum_probabilities > 0:
58          probabilities = [p / sum_probabilities for p in
                probabilities]
59      else:
60          return None
61
62      posibility = random()
63      cumulative_p = np.cumsum(probabilities)
64      chosen_city = np.searchsorted(cumulative_p, posibility)
65
66      return int(chosen_city)
67
68  def ant_alg(matrix, cities, alpha, beta, days, k_evaporation):
69      best_route = []
70      min_len = float("inf")
71      daily_quota = daily_quota_Q(matrix, cities)
72      pheromones = get_matrix_pheromones(cities)
73      visibility = get_matrix_visibility(matrix, cities)
74      ants = cities
75
```

```
76      for day in range(days):
77          route = np.arange(cities)
78          visited = get_matrix_visited(route, ants)
79          for ant in range(ants):
80              while (len(visited[ant]) != ants):
81                  chosen_city = choose_next_city(pheromones, ant,
                        alpha, beta, visibility, visited, cities)
82                  visited[ant].append(chosen_city)     # -1
83
84              cur_len = calc_len_route(matrix, visited[ant])
85              if (cur_len < min_len):
86                  min_len = cur_len
87                  best_route = visited[ant]
88
89          pheromones = update_pheromones(matrix, cities, visited,
                pheromones, daily_quota, k_evaporation)
90
91      return min_len, best_route
```

3.3 Функциональные тесты

В таблице 3.1 приведены тесты для функций программы. Тесты для всех функций пройдены успешно.

Таблица 3.1 – Функциональные тесты

Матрица смежности	Ожидание	Результат
$\begin{pmatrix} 0 & 1 & 7 \\ 1 & 0 & 9 \\ 7 & 9 & 0 \end{pmatrix}$	[1, 0, 2]	[1, 0, 2]
$\begin{pmatrix} 0 & 4 & 4 & 2 \\ 4 & 0 & 1 & 6 \\ 4 & 1 & 0 & 1 \\ 2 & 6 & 1 & 0 \end{pmatrix}$	[0, 3, 2, 1]	[0, 3, 2, 1]

$\begin{pmatrix} 0 & 6 & 3 & 8 & 10 \\ 6 & 0 & 5 & 7 & 8 \\ 3 & 5 & 0 & 2 & 3 \\ 8 & 7 & 2 & 0 & 2 \\ 10 & 8 & 3 & 2 & 0 \end{pmatrix}$	[1, 0, 2, 3, 4]	[1, 0, 2, 3, 4]
$\begin{pmatrix} 0 & 2 & 2 & 8 & 2 & 3 \\ 2 & 0 & 7 & 1 & 1 & 9 \\ 2 & 7 & 0 & 9 & 3 & 5 \\ 8 & 1 & 9 & 0 & 6 & 9 \\ 2 & 1 & 3 & 6 & 0 & 6 \\ 3 & 9 & 5 & 9 & 6 & 0 \end{pmatrix}$	[3, 1, 4, 2, 0, 5]	[3, 1, 4, 2, 0, 5]

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства:

— операционная система Manjaro Linux x86_64;

— процессор Ryzen 5500U 6 ядер, тактовая частота 2.1 ГГц;

— оперативная память 16 Гбайт.

При тестировании ноутбук был включён в сеть электропитания. Во время тестирования ноутбук был нагружен только системными приложениями окружения, а также системой тестирования.

4.2 Сравнительный анализ временных затрат

Замеры времени для каждой размерности матрицы проводились 20 раз. Результат замера — среднее арифметическое время работы алгоритма, на вход подавались матрицы сгенерированные случайным образом.

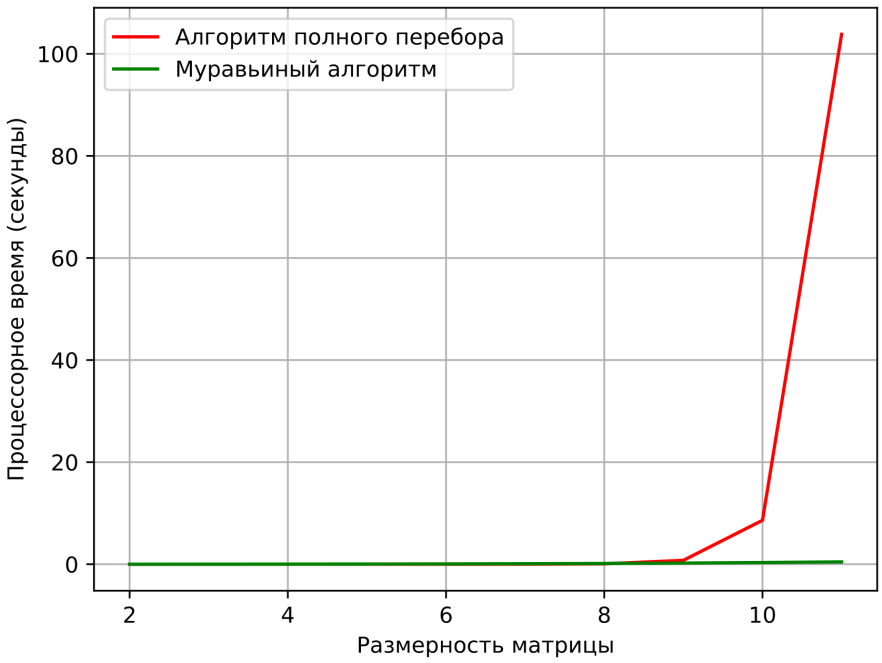

Рисунок 4.1 – Временные затраты

4.3 Результаты проводимых исследований

По полученным данным измерений временных затрат был сделан вывод о том, что алгоритм полного перебора имеет большую вычислительную сложность по сравнению с муравьиным алгоритмом, это хорошо видно при увеличении размерности матрицы. Алгоритм полного перебора является значительно менее эффективным по временным затратам, но его преимуществом является гарантированное нахождение глобального минимума.

4.4 Класс данных 1

Класс данных 1 представляет собой матрицу смежности размерностью 10, разброс длины путей $[1, 10]$.

$$M_1 = \begin{pmatrix} 0 & 6 & 6 & 8 & 8 & 2 & 8 & 7 & 4 & 3 \\ 6 & 0 & 3 & 4 & 7 & 4 & 6 & 9 & 2 & 1 \\ 6 & 3 & 0 & 5 & 9 & 2 & 8 & 4 & 4 & 6 \\ 8 & 4 & 5 & 0 & 4 & 2 & 2 & 1 & 4 & 7 \\ 8 & 7 & 9 & 4 & 0 & 8 & 8 & 4 & 6 & 3 \\ 2 & 4 & 2 & 2 & 8 & 0 & 9 & 5 & 1 & 1 \\ 8 & 6 & 8 & 2 & 8 & 9 & 0 & 3 & 9 & 9 \\ 7 & 9 & 4 & 1 & 4 & 5 & 3 & 0 & 9 & 8 \\ 4 & 2 & 4 & 4 & 6 & 1 & 9 & 9 & 0 & 7 \\ 3 & 1 & 6 & 7 & 3 & 1 & 9 & 8 & 7 & 0 \end{pmatrix}$$

Для данного класса данных при параметризации для каждого набора параметров проводилось 15 измерений результаты приведены в приложении А. Наилучшим набором является набор коэффициентов под номером 10.

4.5 Класс данных 2

Класс данных 2 представляет собой матрицу смежности размерностью 10, разброс длины путей $[1, 1000]$.

$$M_2 = \begin{pmatrix} 0 & 329 & 358 & 948 & 241 & 778 & 103 & 718 & 204 & 114 \\ 329 & 0 & 981 & 614 & 80 & 313 & 19 & 87 & 410 & 539 \\ 358 & 981 & 0 & 277 & 834 & 427 & 100 & 265 & 535 & 818 \\ 948 & 614 & 277 & 0 & 802 & 132 & 34 & 565 & 375 & 91 \\ 241 & 80 & 834 & 802 & 0 & 150 & 224 & 121 & 881 & 879 \\ 778 & 313 & 427 & 132 & 150 & 0 & 240 & 466 & 587 & 154 \\ 103 & 19 & 100 & 34 & 224 & 240 & 0 & 990 & 98 & 1000 \\ 718 & 87 & 265 & 565 & 121 & 466 & 990 & 0 & 833 & 384 \\ 204 & 410 & 535 & 375 & 881 & 5879 & 8 & 833 & 0 & 845 \\ 114 & 539 & 818 & 91 & 879 & 154 & 1000 & 384 & 845 & 0 \end{pmatrix}$$

Для данного класса данных при параметризации для каждого набора параметров проводилось 15 измерений результаты приведены в приложении А. Наилучшим набором является набор коэффициентов под номером 5.

4.6 Класс данных 3

Класс данных 3 представляет собой матрицу смежности размерностью 10, разброс длины путей $[1, 10000]$.

$$M_3 = \begin{pmatrix} 0 & 4886 & 3635 & 2763 & 5266 & 3840 & 6731 & 4223 & 2986 & 8571 \\ 4886 & 0 & 3555 & 8981 & 7726 & 4041 & 7116 & 2701 & 2271 & 4951 \\ 3635 & 3555 & 0 & 169 & 5813 & 7148 & 3570 & 2560 & 6898 & 8612 \\ 2763 & 8981 & 169 & 0 & 472 & 4641 & 2948 & 5613 & 7216 & 4996 \\ 5266 & 7726 & 5813 & 472 & 0 & 2893 & 8984 & 6664 & 3508 & 6807 \\ 3840 & 4041 & 7148 & 4641 & 2893 & 0 & 3981 & 5081 & 1726 & 2477 \\ 6731 & 7116 & 3570 & 2948 & 8984 & 3981 & 0 & 1746 & 7391 & 1641 \\ 4223 & 2701 & 2560 & 5613 & 6664 & 5081 & 1746 & 0 & 4066 & 7988 \\ 2986 & 2271 & 6898 & 7216 & 3508 & 1726 & 7391 & 4066 & 0 & 1177 \\ 8571 & 4951 & 8612 & 4996 & 6807 & 2477 & 1641 & 7988 & 1177 & 0 \end{pmatrix}$$

Для данного класса данных при параметризации для каждого набора параметров проводилось 15 измерений результаты приведены в приложении А. Наилучшим набором является набор коэффициентов под номером 15.

ЗАКЛЮЧЕНИЕ

Исследованы алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжёра. Цель работы достигнута. В ходе выполнения лабораторной работы были решены следующие задачи:

— разработан алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжёра;

— разработано программное обеспечение, для решения задачи коммивояжёра при помощи указанных алгоритмов;

— выполнена параметризация муравьиного алгоритма по трём его параметрам;

— проведён сравнительный анализ алгоритмов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Левитин А. В. Алгоритмы. Введение в разработку и анализ. Москва: Вильямс, 2006. с. 576.

[2] Борознов В. О. Исследование решения задачи коммивояжёра // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2009. № 2. С. 147–151.

[3] Штовба С. Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. 2003. № 4. С. 70–75.