Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н. Э. Баумана» (национальный исследовательский университет)

Дисциплина: «Анализ алгоритмов»

Отчет по лабораторной работе №6

Тема работы: «Задача коммивояжера. Муравьиный алгоритм»

Студент: Волков Е. А.
Группа: ИУ7-55Б
Преподаватели: Волкова Л. Л.,
Строганов Ю. В.

Москва, 2019 г.

Содержание

Введение

Задача коммивояжера занимает особое место в комбинаторной оптимизации и исследовании операций. Она формулируется как задача поиска минимального по стоимости замкнутого маршрута по всем вершинам без повторений на полном взвешенном графе. Содержательно вершины графа являются городами, которые должен посетить коммивояжер, а веса ребер отражают расстояния (длины) или стоимости проезда. Эта задача является NP-трудной, и точный переборный алгоритм её решения имеет факториальную сложность. [4]

Решение данной задачи важно в первую очередь для крупных транспортных компаний, которые стремятся оптимизировать перевозки и минимизировать расходы. [5]

Особый интерес представляет муравьиный алгоритм, способный эффективно находить приближенное решение задачи коммивояжера.

Цель лабораторной работы: изучение подходов к решению задачи коммивояжера на материале алгоритма полного перебора и муравьиного алгоритма.

Задачи работы:

1) изучить муравьиный алгоритм;
2) применить метод динамического программирования для реализации муравьиного алгоритма и полного перебора;
3) экспериментально подтвердить различия во временно́й эффективности алгоритмов при помощи разработанного программного обеспечения на материале замеров процессорного времени;
4) провести параметризацию муравьиного алгоритма;
5) описать и обосновать полученные результаты в отчете о лабораторной работе, выполненного как расчётно-пояснительная записка.

1 Аналитический раздел

В данном разделе будет рассмотрен муравьиный алгоритм.

1.1 Описание алгоритмов

1.1.1 Муравьиный алгоритм

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. [1]

У муравья есть 3 чувства:

1. Обоняние — муравей чует феромон и его концентрацию на ребре.
2. Зрение — муравей оценивает длину ребра.
3. Память — муравей запоминает посещенные города.

При старте матрица феромонов τ инициализируется равномерно некоторой константой.

Если муравей k находится в городе i и выбирает куда пойти, то делает это по вероятностному правилу:

$$P_{ij}(t) = \begin{cases} \dfrac{\tau_{ij}(t)^{\alpha}\eta_{ij}^{\beta}}{\sum_{q=1}^{m}\tau_{iq}(t)^{\alpha}\eta_{iq}^{\beta}}, & \text{если j не посещен,} \\ 0, & \text{иначе,} \end{cases} \tag{1}$$

где

α, β – весовые коэффициенты, которые задают важность феромона и привлекательность ребра, $\alpha + \beta = const$,
$\eta_{iq} = \frac{1}{d_{ij}}$ – привлекательность ребра (города),
d_{ij} – длина ребра.

Кроме того, надо учитывать изменение феромона по формуле 2:

$$\tau(t+1) = \tau_{ij}(t) \cdot (1-\rho) + \sum_{k=1}^{n} \Delta\tau_{k,ij}(t), \tag{2}$$

$$\Delta\tau_{k,ij}(t) = \begin{cases} \dfrac{Q}{L_k}, & \text{если ребро } ij \text{ в маршруте,} \\ 0, & \text{иначе,} \end{cases} \tag{3}$$

где

L_k – длина маршрута k-ого муравья,

ρ – коэффициент испарения феромона,

Q – нормировочная константа порядка длины наилучшего маршрута.

Выводы

Рассмотрен муравьиный алгоритм, выделены ключевые моменты его работы.

2 Конструкторский раздел

В разделе приводится псевдокод муравьиного алгоритма.

2.1 Разработка алгоритмов

Псевдокод муравьиного алгоритма для решения задачи коммивряжера:

1. Ввод матрицы расстояний D
2. Инициализация рёбер - присвоение видимости η_{ij} и начальной концентрации феромона
3. Размещение муравьёв в случайно выбранные города без совпадений
4. Выбор начального кратчайшего маршрута и определение L^*
5. Цикл по времени жизни колонии t=1,tmax
6. Цикл по всем муравьям k=1,m
7. Построить маршрут $T_k(t)$ по правилу (1)
8. 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 согласно (2), (3)
14. Конец цикла по рёбрам
15. Конец цикла по времени
16. Вывести кратчайший маршрут $T*$ и его длину $L*$

На рисунках 1 и 2 приведена функциональная схема муравьиного алгоритма.

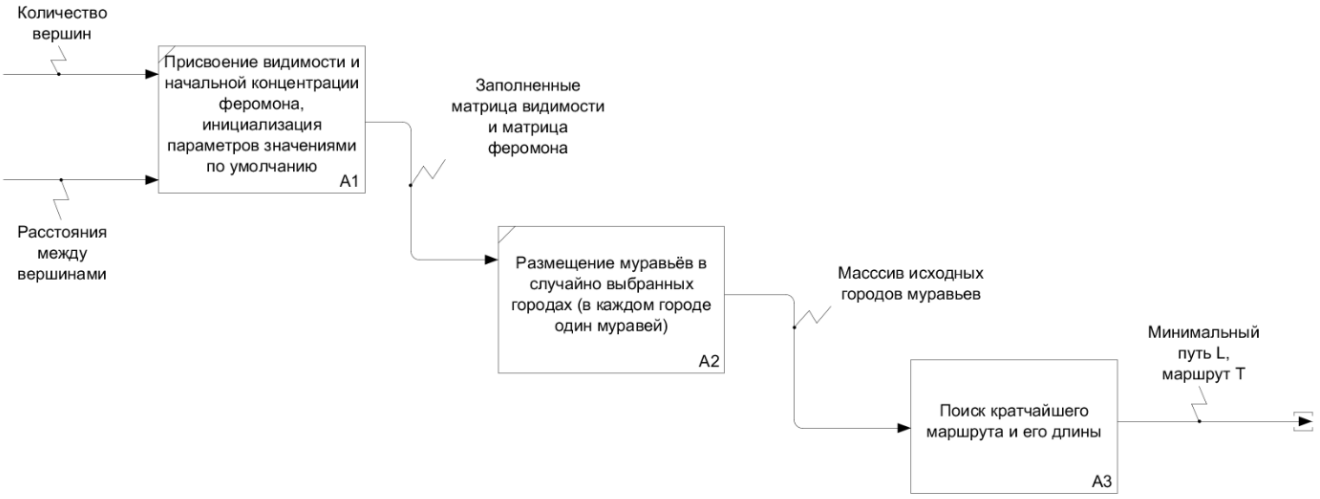

Рис. 1: Функциональная схема муравьиного алгоритма. Основные этапы.

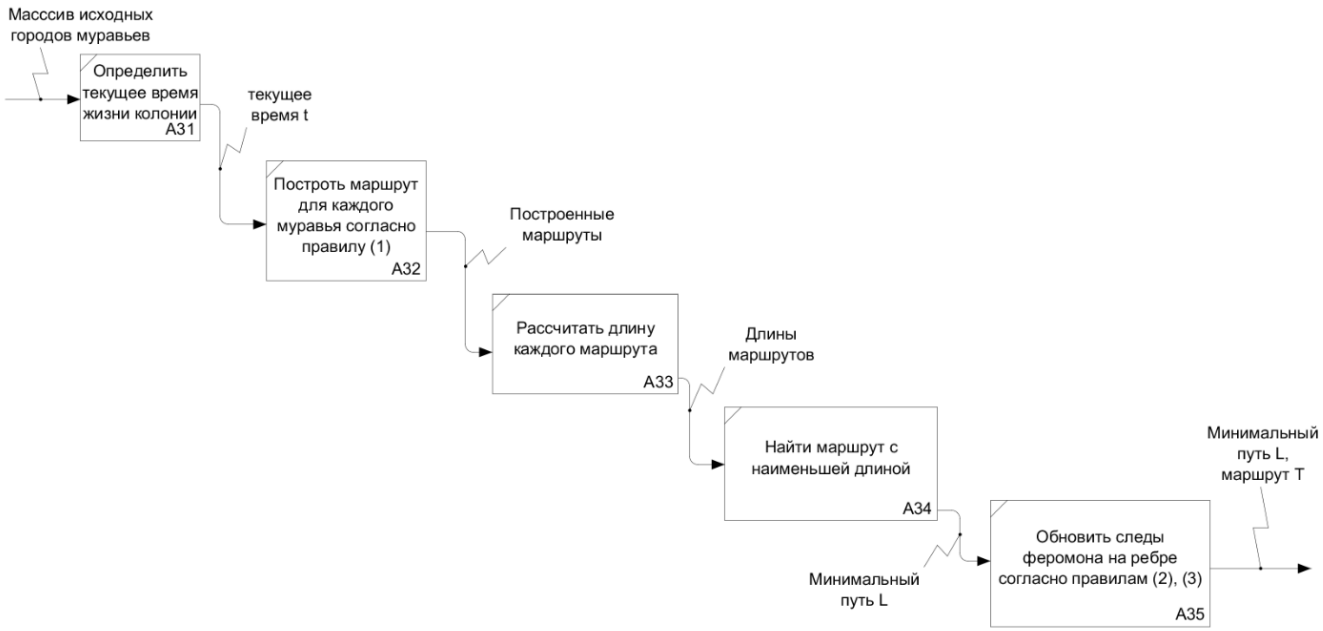

Рис. 2: Функциональная схема муравьиного алгоритма. Поиск кратчайшего пути.

Выводы

В разделе представлен пошаговый разбор муравьиного алгоритма для решения задачи коммивояжера.

3 Технологический раздел

Здесь описываются требования к программному обеспечению и средства реализации и приводятся листинги программы.

3.1 Требования к программному обеспечению

Входные данные:

- количество вершин графа (целое число, большее 2),
- расстояния между вершинами графа (положительные вещественные числа).

Выходные данные: длины кратчайшего пути и маршруты, найденные полным перебором и муравьиным алгоритмом.

Для муравьиного алгоритма по умолчанию заданы параметры
$\alpha = 0.5, \rho = 0.5, t = 300, q = 100$.

На рис. 3 приведена функциональная схема программы решения задачи коммивояжера.

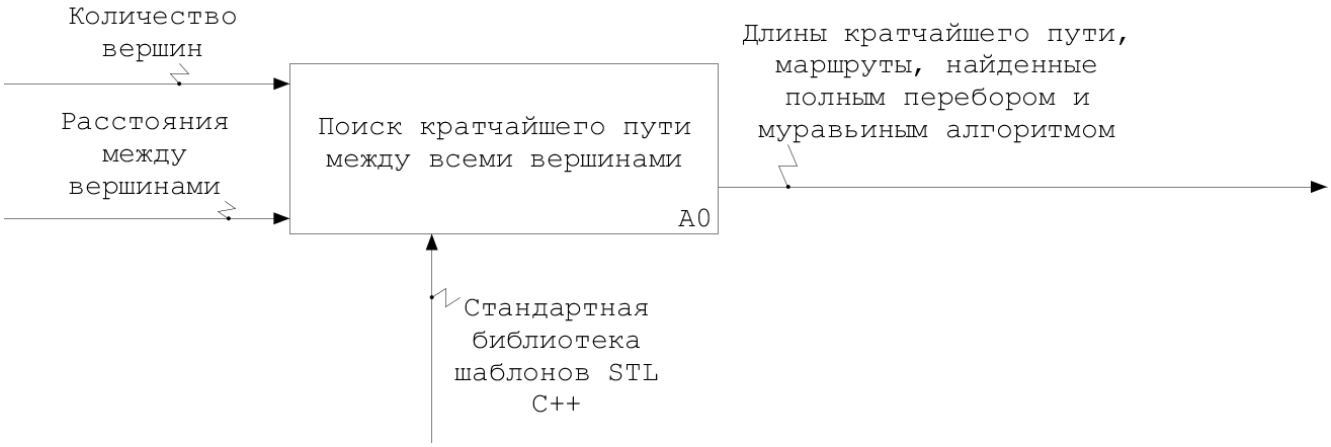

Рис. 3: Функциональная схема программы решения задачи коммивояжера

3.2 Средства реализации

Программа написана на языке C++, т. к. этот язык предоставляет программисту широкие возможности реализации самых разнообразных алгоритмов, обладает высокой эффективностью и значительным набором стандартных классов и процедур. В качестве среды разработки использовался фреймворк QT 5.13.1.

Для хранения массива применяется контейнерный класс *std::vector* из стандартной библиотеки шаблонов STL.

Замер времени выполнения программы производится с помощью библиотеки *chrono*.

3.3 Листинг программы

Реализованная программа представлена в листингах 1-2.

Листинг 1: Реализация полного перебора для решения задачи коммивояжера

```
double total_search(Matrix matrix, std::vector<int> &way) {
    int n = matrix.size();
    double min_len = -1;

    std::vector<int> p(n);  // permutation
    for (int i = 0; i < n; ++i) {
        p[i] = i;
    }

    do {
        int len = 0;

        for (int i = 0; i < n - 1; ++i) {
            len += matrix[p[i]][p[i + 1]];
        }

        len += matrix[p[n - 1]][p[0]];
        if (len < min_len || min_len < 0) {
            min_len = len;
            for (int i = 0; i < n; ++i) {
                way[i] = p[i];
            }
        }
    } while(std::next_permutation(p.begin(), p.end()));

    return min_len;
}
```

Листинг 2: Реализация муравьиного алгоритма для решения задачи коммивояжера

```
std::random_device rd;
std::mt19937 g(rd());

double probability(int to, int size, int n,
    const std::vector<int> &ant,
    const Matrix &pheromone, const Matrix &dist,
    double alpha, double betta) {
    for (int i = 0; i < size; ++i) {
        if (to == ant[i]) {
            return 0;
        }
    }

```

```
14      double sum = 0.0;
15      int from = ant[size - 1];
16
17      for (int j = 0; j < n; ++j) {
18          bool flag = true;
19
20          for (int i = 0; i < size; ++i) {
21              if (j == ant[i]) {
22                  flag = false;
23              }
24          }
25
26          if (flag) {
27              sum +=
28                  pow(pheromone[from][j], alpha) * pow(dist[from][j], betta);
29          }
30      }
31
32      return
33          pow(pheromone[from][to], alpha) * pow(dist[from][to], betta) / sum;
34  }
35
36  double aco(Matrix matrix, std::vector<int> &way,
37      double alpha, double rho, int t_max) {
38      double betta = 1 - alpha;
39      int n = matrix.size();
40      int m = n;
41      double min_len = -1;
42
43      Matrix dist(n, std::vector<double>(n));
44      Matrix pheromone(n, std::vector<double>(n, 1.0 / n));
45
46      for (int i = 0; i < n; ++i) {
47          for (int j = 0; j < n; ++j) {
48              if (i != j) {
49                  dist[i][j] = 1.0 / matrix[i][j];
50              }
51          }
52      }
53
54      std::vector<std::vector<int>> ants(m, std::vector<int>(n, -1));
55      std::vector<int> starts(n);
56      for (int i = 0; i < n; ++i) {
57          starts[i] = i;
58      }
59
60      for (int i = 0; i < m; ++i) {
61          if (0 == i % n) {
62              std::shuffle(starts.begin(), starts.end(), g);
63          }
64
65          ants[i][0] = starts[i % n];
66      }
67
68      std::vector<double> len(m, 0);
```

```
69
70      for (int t = 0; t < t_max; ++t) {
71          for (int k = 0; k < m; ++k) {
72              for (int i = 1; i < n; ++i) {
73                  int j_max = -1;
74                  double p_max = 0.0;
75
76                  for (int j = 0; j < n; ++j) {
77                      if (ants[k][i - 1] != j) {
78                          double p = probability(
79                              j, i, n, ants[k], pheromone, dist, alpha, betta
80                          );
81                          if (p && p >= p_max) {
82                              p_max = p;
83                              j_max = j;
84                          }
85                      }
86                  }
87
88                  len[k] += matrix[ants[k][i - 1]][j_max];
89                  if (i == n - 1) {
90                      len[k] += matrix[j_max][ants[k][0]];
91                  }
92
93                  ants[k][i] = j_max;
94              }
95
96              for (int i = 0; i < n; ++i) {
97                  int from = ants[k][i % n];
98                  int to = ants[k][(i + 1) % n];
99
100                 pheromone[from][to] += Q / len[k];
101                 pheromone[to][from] = pheromone[from][to];
102             }
103
104             if (len[k] < min_len || min_len < 0) {
105                 min_len = len[k];
106                 for (int i = 0; i < n; ++i) {
107                     way[i] = ants[k][i];
108                 }
109             }
110
111             len[k] = 0;
112         }
113
114         for (int i = 0; i < n; ++i) {
115             for (int j = 0; j < n; ++j) {
116                 if (i != j) {
117                     pheromone[i][j] *= (1 - rho);
118                 }
119             }
120         }
121     }
122
123     return min_len;
```

```
124 }
```

Выводы

В данном разделе были рассмотрены требования к программному обеспечению, обоснован выбор средств реализации, приведены листинги программы.

4 Исследовательский раздел

В разделе представлены примеры выполнения программы, результаты сравнения алгоритмов решения задачи коммивояжера, а также исследование эффективности поиска муравьиным алгоритмом при различных параметрах.

4.1 Примеры работы

На рис. 4-5 приведены примеры работы программы.

```
Input number of vertices : sdfsdf
Incorrect input!
Input number of vertices : -10
Incorrect input!
Input number of vertices : 3
Input distances :
input 0 rows: dfsdf 0 3 5
Incorrect input!
 input 0 rows: 1 2 3 4
Incorrect input!
 input 0 rows: 1 f 3
Incorrect input!
 input 0 rows: 1 2 3
Incorrect input!
 input 0 rows: 0 1 2
input 1 rows: 4 5 6
Incorrect input!
 input 1 rows: 4 0 5
input 2 rows: 6 7 8
Incorrect input!
 input 2 rows: 6 7 0
```

Рис. 4: Пример работы программы для некорректного ввода

```
Input number of vertices : 4
Input distances :
input 0 rows: 0 1 2 3
input 1 rows: 4 0 5 6
input 2 rows: 7 8 0 9
input 3 rows: 10 11 12 0

Matrix distances:
0 1 2 3
4 0 5 6
7 8 0 9
10 11 12 0

TOTAL SEARCH
Way length : 25
Way : ( 1 -> 2 -> 3 -> 4 )

ACO
Way length : 25
Way : ( 4 -> 1 -> 2 -> 3 )
```

Рис. 5: Пример работы программы для корректного ввода

4.2 Постановка эксперимента

1. Сравнить время работы полного перебора и муравьиного алгоритма на 5 матрицах размерностью 10, замер каждого вычисления проводится 100 раз. Параметры: $\alpha = 0.5, \rho = 0.5, t = 300, q = 100$.
2. Выяснить при каких параметрах $\alpha \in [0; 1]$, $\rho \in (0; 1]$, $t \in [10; 200]$, где $\alpha, \rho \in \mathbb{R}$, $t \in \mathbb{N}$ муравьиный алгоритм будет работать лучше. При этом значения α и ρ меняются с шагом 0.1, t — с шагом 10.

4.3 Сравнительный анализ на основе эксперимента

4.3.1 Сравнение времени работы

Замеры производились на компьютере следующей конфигурации:

1) процессор: Intel Core i5 1.8 ГГц, 8 логических ядер;
2) ОЗУ: 8 Гб, 2400 МГц DDR4;
3) ОС: Windows 10.

В таблице 1 содержатся замеры времени для алгоритма полного перебора и алгоритма муравьев на пяти тестовых матрицах.

Таблица 1: Сравнение времени выполнения алгоритмов решения задачи коммивояжера

Матрица	Перебор, с	Муравьи, с
1	1,39128	0,404246
2	1,37277	0,538537
3	1,36178	0,673532
4	1,36118	0,80861
5	1,35346	0,942806

Как и ожидалось, муравьиный алгоритм быстрее решает поставленную задачу — на экспериментальных данных в среднем затрачено в 2 раза меньше времени, чем при полном переборе.

4.3.2 Параметризация в муравьином алгоритме

Для проведения параметризации в процессе случайной генерации получены 5 матриц размерностью 10, элементы матрицы находятся в диапазоне от 0 до 100 и кратны 10.

1. Путь: $(1 \to 8 \to 3 \to 10 \to 2 \to 5 \to 6 \to 4 \to 7 \to 9)$

Минимальная длина: 130

$$\begin{bmatrix} 0 & 50 & 70 & 90 & 10 & 30 & 70 & 10 & 60 & 90 \\ 50 & 0 & 60 & 60 & 10 & 80 & 40 & 10 & 80 & 80 \\ 50 & 80 & 0 & 10 & 70 & 70 & 50 & 10 & 70 & 10 \\ 10 & 80 & 70 & 0 & 30 & 30 & 40 & 80 & 40 & 40 \\ 40 & 20 & 50 & 20 & 0 & 10 & 10 & 70 & 20 & 20 \\ 10 & 60 & 70 & 10 & 60 & 0 & 40 & 70 & 20 & 50 \\ 80 & 80 & 60 & 90 & 20 & 70 & 0 & 40 & 10 & 60 \\ 50 & 30 & 10 & 60 & 50 & 10 & 50 & 0 & 10 & 90 \\ 10 & 70 & 90 & 30 & 10 & 50 & 40 & 10 & 0 & 40 \\ 90 & 10 & 50 & 10 & 10 & 20 & 80 & 90 & 20 & 0 \end{bmatrix}$$

2. Путь: $(1 \to 7 \to 3 \to 5 \to 9 \to 10 \to 8 \to 4 \to 6 \to 2)$

   Минимальная длина: 170

$$\begin{bmatrix} 0 & 60 & 30 & 90 & 90 & 70 & 30 & 80 & 20 & 40 \\ 10 & 0 & 10 & 50 & 70 & 70 & 20 & 20 & 60 & 40 \\ 80 & 20 & 0 & 60 & 10 & 40 & 60 & 80 & 20 & 80 \\ 70 & 80 & 50 & 0 & 40 & 20 & 50 & 30 & 30 & 90 \\ 90 & 20 & 10 & 80 & 0 & 50 & 80 & 70 & 20 & 30 \\ 30 & 30 & 70 & 70 & 70 & 0 & 90 & 10 & 90 & 80 \\ 30 & 30 & 10 & 60 & 30 & 50 & 0 & 30 & 70 & 10 \\ 70 & 60 & 30 & 10 & 10 & 50 & 50 & 0 & 90 & 80 \\ 60 & 50 & 30 & 60 & 60 & 60 & 60 & 20 & 0 & 10 \\ 50 & 70 & 80 & 40 & 60 & 70 & 70 & 20 & 30 & 0 \end{bmatrix}$$

3. Путь: $(1 \to 7 \to 3 \to 4 \to 6 \to 5 \to 9 \to 2 \to 8 \to 10)$

   Минимальная длина: 180

$$\begin{bmatrix} 0 & 70 & 30 & 20 & 90 & 40 & 10 & 40 & 10 & 70 \\ 70 & 0 & 30 & 10 & 80 & 80 & 20 & 20 & 40 & 30 \\ 50 & 70 & 0 & 10 & 10 & 90 & 10 & 70 & 10 & 40 \\ 50 & 70 & 60 & 0 & 10 & 20 & 70 & 50 & 60 & 20 \\ 30 & 80 & 10 & 90 & 0 & 10 & 10 & 20 & 10 & 30 \\ 50 & 80 & 30 & 70 & 10 & 0 & 50 & 90 & 50 & 30 \\ 10 & 50 & 10 & 30 & 10 & 20 & 0 & 70 & 80 & 30 \\ 20 & 70 & 10 & 30 & 30 & 30 & 90 & 0 & 10 & 20 \\ 80 & 50 & 60 & 80 & 90 & 60 & 70 & 70 & 0 & 80 \\ 20 & 90 & 20 & 20 & 20 & 90 & 50 & 80 & 90 & 0 \end{bmatrix}$$

4. Путь: $(1 \to 3 \to 2 \to 8 \to 5 \to 6 \to 10 \to 4 \to 9 \to 7)$

Минимальная длина: 110

$$\begin{bmatrix} 0 & 10 & 10 & 10 & 80 & 70 & 40 & 20 & 10 & 10 \\ 20 & 0 & 60 & 70 & 70 & 80 & 40 & 10 & 90 & 10 \\ 10 & 10 & 0 & 50 & 70 & 90 & 50 & 40 & 20 & 40 \\ 60 & 10 & 10 & 0 & 60 & 90 & 50 & 60 & 10 & 20 \\ 60 & 60 & 60 & 20 & 0 & 10 & 10 & 10 & 30 & 70 \\ 60 & 60 & 80 & 10 & 90 & 0 & 70 & 80 & 90 & 10 \\ 10 & 70 & 30 & 60 & 30 & 20 & 0 & 50 & 10 & 30 \\ 40 & 10 & 80 & 80 & 10 & 50 & 60 & 0 & 70 & 30 \\ 60 & 20 & 70 & 30 & 20 & 80 & 20 & 60 & 0 & 20 \\ 20 & 20 & 90 & 10 & 10 & 50 & 20 & 50 & 70 & 0 \end{bmatrix}$$

5. Путь: $(1 \to 3 \to 4 \to 10 \to 7 \to 8 \to 2 \to 6 \to 9 \to 5)$

   Минимальная длина: 160

$$\begin{bmatrix} 0 & 90 & 10 & 40 & 10 & 90 & 90 & 70 & 60 & 60 \\ 70 & 0 & 90 & 50 & 40 & 10 & 40 & 90 & 50 & 50 \\ 10 & 30 & 0 & 50 & 20 & 10 & 90 & 60 & 50 & 20 \\ 70 & 60 & 90 & 0 & 60 & 30 & 60 & 10 & 20 & 20 \\ 10 & 10 & 90 & 60 & 0 & 30 & 70 & 40 & 60 & 50 \\ 20 & 40 & 80 & 60 & 60 & 0 & 30 & 80 & 10 & 70 \\ 50 & 40 & 10 & 50 & 40 & 60 & 0 & 10 & 40 & 30 \\ 60 & 20 & 50 & 80 & 70 & 70 & 50 & 0 & 10 & 40 \\ 30 & 60 & 30 & 80 & 10 & 10 & 40 & 30 & 0 & 80 \\ 60 & 30 & 10 & 90 & 20 & 30 & 10 & 20 & 10 & 0 \end{bmatrix}$$

В таблицах 2-4 приведены соответствия параметров с полученными результатами для матрицы 1, 2 и 3. Для остальных матриц были получены аналогичные таблицы.

Таблица 2: Параметризация на матрице 1

№	α	ρ	t	Ответ	Муравьи
0	0.0	0.1	10	130	190
1	0.0	0.1	20	130	190
2	0.0	0.1	30	130	190
3	0.0	0.1	40	130	190
4	0.0	0.1	50	130	190
5	0.0	0.1	60	130	190
...	...	...	...	...	...
360	0.1	0.9	10	130	170
361	0.1	0.9	20	130	170
362	0.1	0.9	30	130	160
363	0.1	0.9	40	130	160
364	0.1	0.9	50	130	160
365	0.1	0.9	60	130	170
...	...	...	...	...	...
806	0.4	0.1	70	130	160
807	0.4	0.1	80	130	150
808	0.4	0.1	90	130	150
809	0.4	0.1	100	130	170
810	0.4	0.1	110	130	150
...	...	...	...	...	...
1517	0.7	0.6	180	130	190
1518	0.7	0.6	190	130	190
1519	0.7	0.6	200	130	190
1520	0.7	0.7	10	130	190
1521	0.7	0.7	20	130	190
1522	0.7	0.7	30	130	190
...	...	...	...	...	...
2195	1.0	1.0	160	130	270
2196	1.0	1.0	170	130	300
2197	1.0	1.0	180	130	270
2198	1.0	1.0	190	130	280
2199	1.0	1.0	200	130	280

Таблица 3: Параметризация на матрице 2

№	α	ρ	t	Ответ	Муравьи
0	0.0	0.1	10	170	170
1	0.0	0.1	20	170	170
2	0.0	0.1	30	170	170
3	0.0	0.1	40	170	170
4	0.0	0.1	50	170	170
5	0.0	0.1	60	170	170
...	...	...	...	...	...
360	0.1	0.9	10	170	170
361	0.1	0.9	20	170	170
362	0.1	0.9	30	170	170
363	0.1	0.9	40	170	170
364	0.1	0.9	50	170	170
365	0.1	0.9	60	170	170
...	...	...	...	...	...
806	0.4	0.1	70	170	170
807	0.4	0.1	80	170	170
808	0.4	0.1	90	170	170
809	0.4	0.1	100	170	190
810	0.4	0.1	110	170	190
...	...	...	...	...	...
1517	0.7	0.6	180	170	190
1518	0.7	0.6	190	170	210
1519	0.7	0.6	200	170	190
1520	0.7	0.7	10	170	170
1521	0.7	0.7	20	170	170
1522	0.7	0.7	30	170	170
...	...	...	...	...	...
2195	1.0	1.0	160	130	290
2196	1.0	1.0	170	130	290
2197	1.0	1.0	180	130	290
2198	1.0	1.0	190	130	290
2199	1.0	1.0	200	130	290

Таблица 4: Параметризация на матрице 3

№	α	ρ	t	Ответ	Муравьи
0	0.0	0.1	10	180	210
1	0.0	0.1	20	180	210
2	0.0	0.1	30	180	210
3	0.0	0.1	40	180	210
4	0.0	0.1	50	180	210
5	0.0	0.1	60	180	210
...	...	...	...	...	...
360	0.1	0.9	10	180	200
361	0.1	0.9	20	180	190
362	0.1	0.9	30	180	200
363	0.1	0.9	40	180	190
364	0.1	0.9	50	180	200
365	0.1	0.9	60	180	190
...	...	...	...	...	...
806	0.4	0.1	70	180	190
807	0.4	0.1	80	180	190
808	0.4	0.1	90	180	210
809	0.4	0.1	100	180	200
810	0.4	0.1	110	180	200
...	...	...	...	...	...
1517	0.7	0.6	180	180	210
1518	0.7	0.6	190	180	210
1519	0.7	0.6	200	180	210
1520	0.7	0.7	10	180	210
1521	0.7	0.7	20	180	210
1522	0.7	0.7	30	180	210
...	...	...	...	...	...
2195	1.0	1.0	160	180	350
2196	1.0	1.0	170	180	340
2197	1.0	1.0	180	180	340
2198	1.0	1.0	190	180	280
2199	1.0	1.0	200	180	350

Значения α, ρ, t, на которых муравьиный алгоритм показал приемлемый результат для всех 5 выбранных матриц, представлены в таблице 5.

Таблица 5: Результаты параметризации

α	ρ	t
0.1	0.1	60
0.1	0.2	50
0.1	0.2	180
0.1	0.3	170
0.1	0.9	50
0.1	0.9	150
0.1	0.9	190
0.2	0.2	80
0.2	0.2	200
0.2	0.3	60
0.2	0.3	140
0.2	0.4	190
0.2	0.5	90
0.2	0.5	100
0.2	0.6	90
0.2	0.6	140
0.2	0.6	200
0.2	0.7	10
0.2	0.7	190
0.2	0.8	60
0.2	0.8	180
0.2	0.9	190
0.2	0.9	200
0.3	0.1	140
0.3	0.3	10
0.3	0.9	160
0.3	1.0	110
0.4	0.1	60
0.4	0.1	160
0.4	0.3	110
0.4	0.5	40
0.4	0.6	160
0.4	0.8	60
0.4	1.0	70

Выводы

Как видно из результатов, хорошие результаты для муравьиного алгоритма получены при параметрах $\alpha \in [0,1; 0,4]$, $\rho \in (0; 1]$ и $t \in [10; 200]$. Однако, стоит учесть, что для других классов задач эти параметры не дадут должного результата. Для каждого отдельного случая необходимо проводить параметризацию заново.

Заключение

В работе экспериментально подтверждена эффективность муравьиного алгоритма в сравнении с точным перебором всех маршрутов (на матрицах размерностью 10 быстрее приблизительно в 2 раза). Также проведена параметризация вероятностного алгоритма муравья. Из результатов видно, что при $\alpha \in [0,1; 0,4]$ в сочетании с $\rho \in (0; 1]$ и $t \in [10; 200]$ получаются приемлемые результаты на случайных матрицах размерностью 10. Однако, стоит учесть, что для других классов задач эти параметры не дадут должного результата. Для каждого отдельного случая необходимо проводить параметризацию заново.

Список литературы

[1] Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях, 2003, №4, с.70-75

[2] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход.- М.:Техносфера, 2009.

[3] Д. Кнут. Искусство программирования, М., Мир, 1978

[4] Задача коммивояжера [Электронный ресурс]. – Режим доступа: http://www.math.nsc.ru/LBRT/k5/OR-MMF/TSPr.pdf, свободный – (02.12.2019)

[5] Практическое применение алгоритма решения задачи коммивояжера [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/prakticheskoe-primenenie-algoritma-resheniya-zadachi-kommivoyazhera/, свободный – (01.12.2019)

[6] Алгоритмы муравья [Электронный ресурс]. – Режим доступа: http://www.berkut.mk.ua/download/pdfsmyk/algorMurav.pdf, свободный – (24.11.2019)

[7] Оптимизация методом колонии муравьев [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/163887/, свободный – (28.11.2019)