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»

Отчет по лабораторной работе №6 по курсу "Анализ алгоритмов"

Тема Муравьиный алгоритм

Студент Козлова И.В.

Группа ИУ7-52Б

Оценка (баллы) _____

Преподаватель Волкова Л.Л.

Москва — 2021 г.

Содержание

Введение

Муравьиный алгоритм – алгоритм оптимизации подражанием муравьиной колонии. Один из эффективных полиномиальных алгоритмов для нахождения приближённых решений задачи коммивояжёра, а также решения аналогичных задач поиска маршрутов на графах.

Муравьиные алгоритмы серьезно исследуются европейскими учеными с середины 1990-х годов. Муравьиный алгоритм предоставляет хорошие результаты оптимизации для многих сложных комбинаторных задач, таких как:

- задачи коммивояжера;
- задачи оптимизации маршрутов грузовиков;
- задачи календарного планирования;
- задачи оптимизации сетевых графиков.

Целью данной лабораторной работы является реализация муравьиного алгоритма и приобретение навыков параметризации методов на примере реализованного алгоритма, примененного к задаче коммивояжера.

Для достижения данной цели необходимо решить следующие задачи.

1. Изучить алгоритм полного перебора для решения задачи коммивояжера.
2. Реализовать алгоритм полного перебора для решения задачи коммивояжера.
3. Изучить муравьиный алгоритм для решения задачи коммивояжера.
4. Реализовать муравьиный алгоритм для решения задачи коммивояжера.
5. Провести параметризацию муравьиного алгоритма на трех классах данных.
6. Провести сравнительный анализ скорости работы реализованных алгоритмов.

7. Описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В данном разделе представлены теоретические сведения о рассматриваемых алгоритмах.

1.1 Задача коммивояжера

Коммивояжёр (фр. commis voyageur) — бродячий торговец. Задача коммивояжёра — одна из самых важных задач транспортной логистики, отрасли, занимающейся планированием транспортных перевозок [1]. В описываемой задаче рассматривается несколько городов и матрица попарных расстояний между ними (матрица смежности, если рассматривать граф).

Требуется найти такой порядок посещения городов, чтобы суммарное пройденное расстояние было минимальным, каждый город посещался ровно один раз и коммивояжер вернулся в тот город, с которого начал свой маршрут. Другими словами, во взвешенном полном графе требуется найти гамильтонов цикл минимального веса [2].

1.2 Алгоритм полного перебора для решения задачи коммивояжера

Алгоритм полного перебора для решения задачи коммивояжера предполагает рассмотрение всех возможных путей в графе и выбор наименьшего из них. Смысл перебора состоит в том, что мы перебираем все варианты объезда городов и выбираем оптимальный. Однако, при таком подходе количество возможных маршрутов очень быстро возрастает с ростом n (сложность алгоритма равна $n!$).

Такой подход гарантирует точное решение задачи, однако, уже при небольшом числе городов решение за приемлемое количество времени невозможно.

1.3 Муравьиный алгоритм для решения задачи коммивояжера

Муравьиные алгоритмы представляют собой перспективный метод решения задач коммивояжера, в основе которого лежит моделирование поведения колонии муравьев [3].

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.

Самоорганизация колонии является результатом взаимодействия следующих компонентов:

- случайность — муравьи имеют случайную природу движения (как будто бросили монетку);
- многократность – колония допускает число муравьев, достигающее от нескольких десятков до миллионов особей;
- положительная обратная связь – во время движения муравей откладывает феромон, позволяющий другим особям определить для себя оптимальный маршрут;
- отрицательная обратная связь – по истечении определенного времени феромон испаряется.

Пусть каждый из муравьев обладает следующими характеристиками:

- память - запоминает маршрут, который прошел;
- зрение - определяет длину ребра;
- обоняние - чувствует феромон.

Введем некую целевую функцию (1.1)

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

где D_{ij} — расстояние из текущего пункта i до заданного пункта j.

Посчитаем вероятности перехода в заданную точку по формуле (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 ij.

Когда все муравьи завершили движение происходит обновление феромона по формуле (1.3):

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

При этом

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

где

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

где

- L_k — длина пути k-ого муравья;
- N — количество муравьев;
- Q — настраивает концентрацию нанесения/испарения феромона.

Описание поведения муравьев при выборе пути.

1. Муравьи имеют собственную «память». Поскольку каждый город может быть посещён только один раз, то у каждого муравья есть список

уже посещенных городов - список запретов. Обозначим через J_{ik} список городов, которые необходимо посетить муравью k, находящемуся в городе i.

2. Муравьи обладают «зрением» - желание посетить город j, если муравей находится в городе i. Будем считать, что видимость обратно пропорциональна расстоянию между городами.

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 $\tau_{i,j}(t)$.

4. 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 муравьем 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 формулой (1.5).

1.4 Вывод

В данном разделе были рассмотрены основополагающие материалы, которые в дальнейшем потребуются для реализации алгоритмов. Были рассмотрены задача коммивояжера, муравьиный алгоритм и алгоритм полного перебора.

2 Конструкторская часть

В данном разделе представлены схемы муравьиного алгоритма и алгоритма полного перебора для решения задачи коммивояжера.

2.1 Требования к вводу

Ниже описанные требования необходимо соблюдать для верной работы программы.

1. При запуске программы в командной строке необходимо прописать 1 или 0 (это является выбором режима, 0 – работа с пользователем, 1 – режим тестирования, записи в лог файл);

2. Если на вход в командной программе подается 1, то необходимо также ввести имя файла с данными.

2.2 Разработка алгоритмов

2.2.1 Алгоритм полного перебора

Схема алгоритма полного перебора для решения задачи коммивояжера представлена на рисунке 2.1.

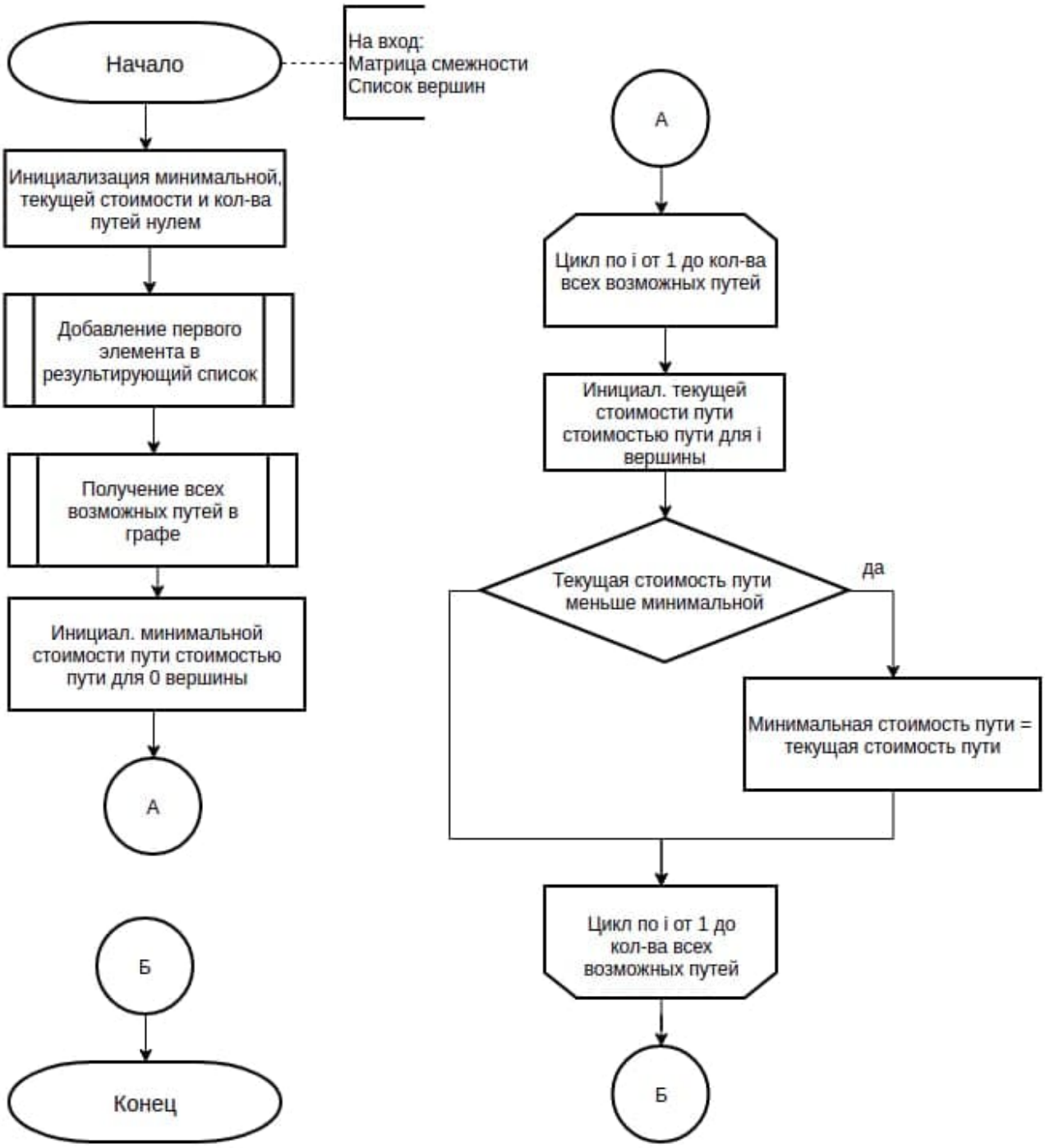

Рисунок 2.1 – Схема алгоритма полного перебора

Схема алгоритма нахождения всех перестановок в графе, использующаяся в алгоритме полного перебора, предоставлена на рисунке 2.2.

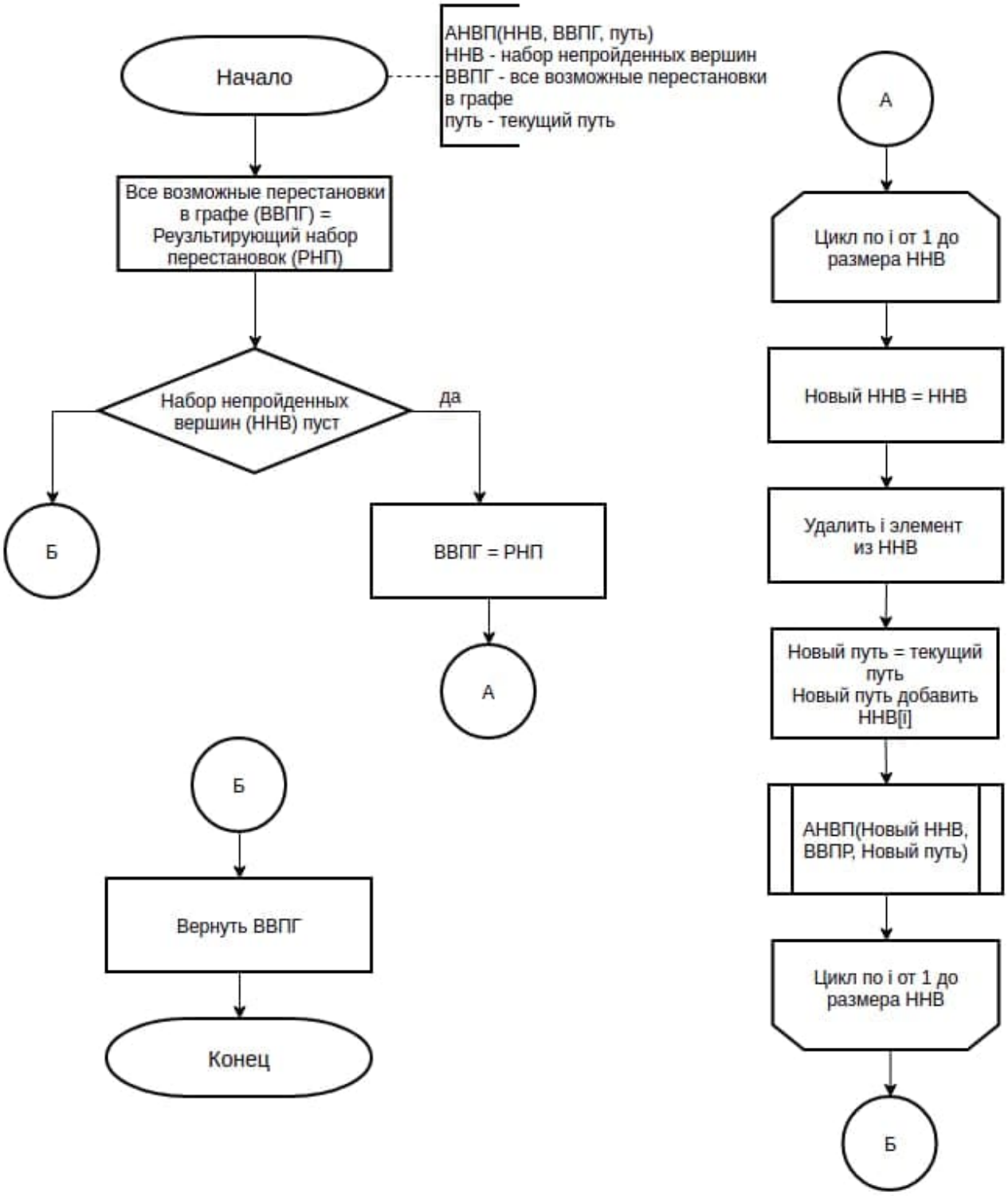

Рисунок 2.2 – Схема алгоритма нахождения всех перестановок в графе

2.2.2 Муравьиный алгоритм

Схема реализации муравьиного алгоритма для решения задачи коммивояжера представлена на рисунке 2.3. Необходимые параметры – матрица феромонов, значение минимального пути.

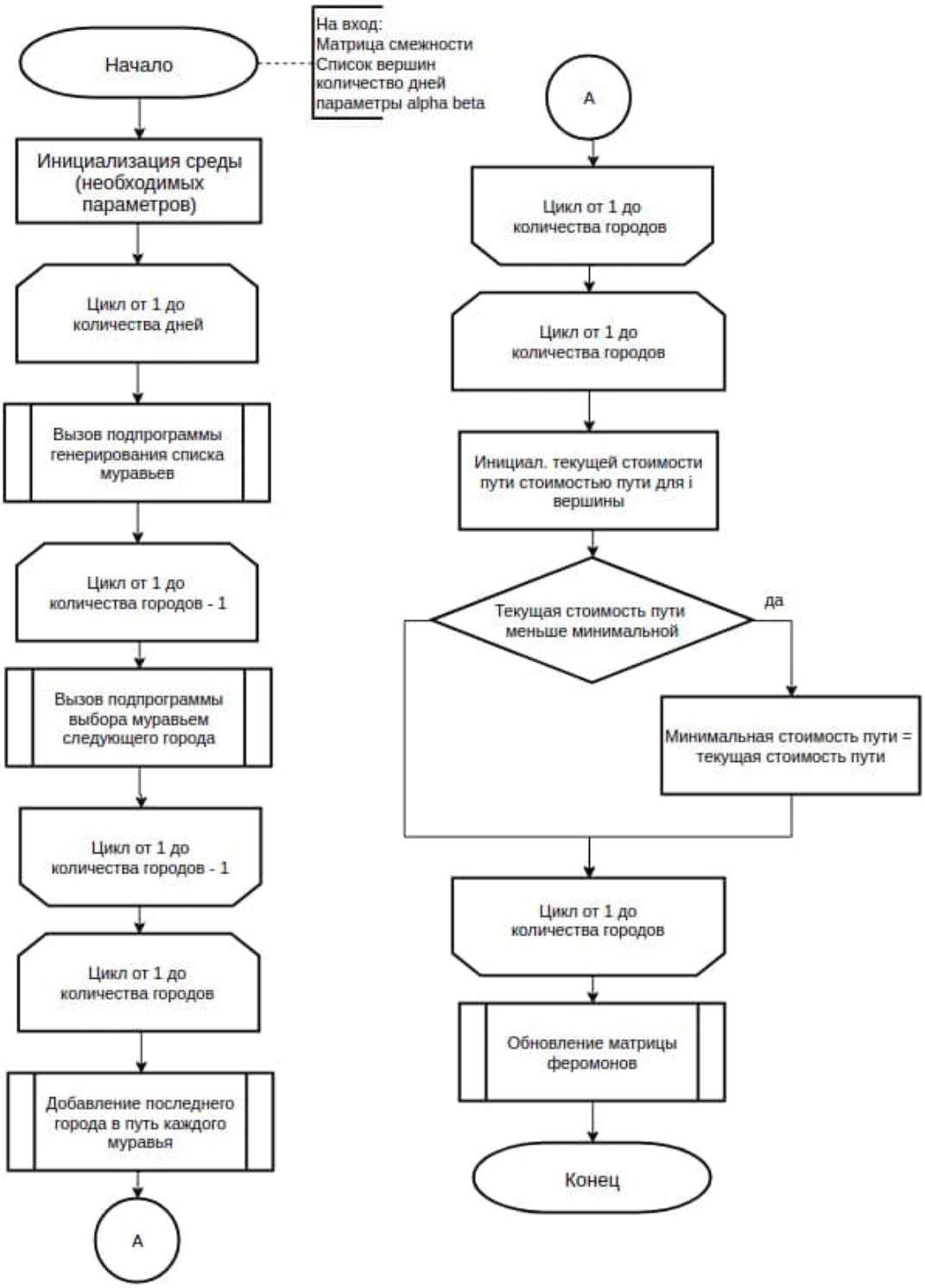

Рисунок 2.3 – Схема реализации муравьиного алгоритма

Схема алгоритма выбора оптимального пути для муравья представлена на рисунке **??**.

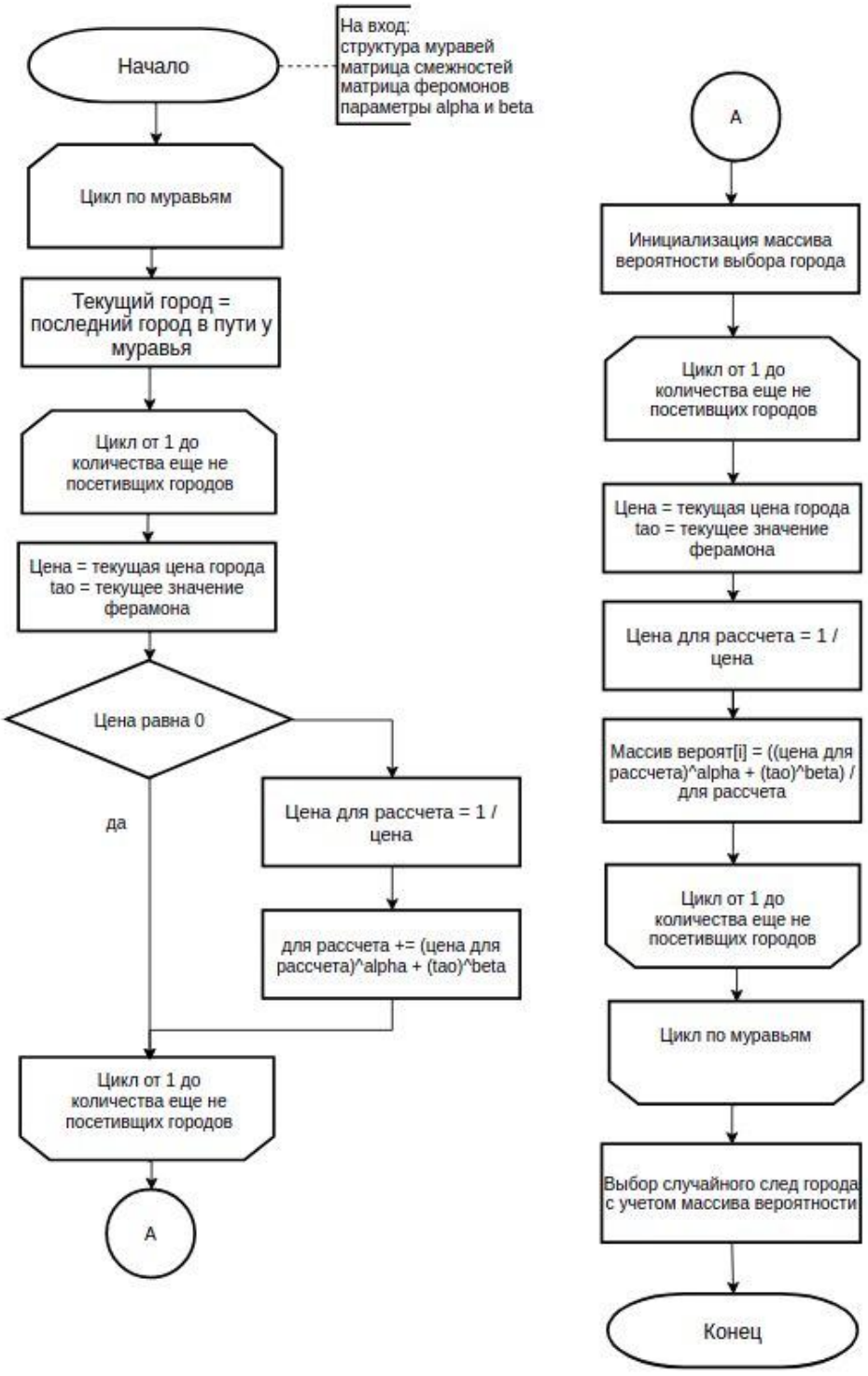

Рисунок 2.4 – Схема алгоритма выбора оптимального пути для муравья

2.3 Вывод

В данном разделе были рассмотрены схемы алгоритма полного перебора и муравьиного алгоритма.

3 Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Выбор языка программирования

В данной лабораторной работе использовался язык программирования - C [4]. Данный язык предоставляет удобные инструменты для работы со структурой данных и обеспечивает хорошую производительность программного продукта.

В качестве среды разработки выбор сделан в сторону Visual Studio Code [5]. Данная среда подходит как для Windows, так и для Linux.

3.2 Сведения о модулях программы

Программа состоит из одного модуля

- main.c – главный файл программы, в которых располагается точка входа;
- bforce.c – файл для реализации алгоритма полного перебора;
- ant_algoritm.c – файл для реализации муравьиного алгоритма;
- array.c – файл для организации списка (массива);
- matrix.c – файл для работы с матрицей смежности;
- city.c – файл для организации списка городов.

3.3 Реализация алгоритмов

В листинге 3.1 представлен алгоритм полного перебора.

Листинг 3.1 – Реализация алгоритма полного перебора

```
sarray get_shortest_path(sarray *cities, int matrix[N][N])
{
    sarray result[MAXRECURSION];
    sarray res_;

    int min_index = 0;
    int min_cost;
    int cur_cost;

    int len_routes = 0;

    del_element(cities, 0);
    add_element(&res_, 0);

    get_routes(cities, &res_, result, &len_routes);
    min_cost = get_cost(result[min_index], matrix);

    for (int i = 1; i < len_routes; i++)
    {
        cur_cost = get_cost(result[i], matrix);
        if (cur_cost < min_cost)
        {
            min_cost = cur_cost;
            min_index = i;
        }
    }
    return result[min_index];
}
```

В листинге 3.2 представлен алгоритм нахождения всех перестановок в графе.

Листинг 3.2 – Реализация алгоритма нахождения всех перестановок в графе

```
void get_routes(sarray *cities, sarray *res_,
sarray result[MAXRECURSION], int *len)
{
  int element;
  sarray buf;

  if (!cities->len)
```

```
  {
    add_element(res_, get_element(*res_, 0));
    result[*len] = *res_;
    (*len)++;
    del_element(res_, res_->len - 1);
  }

  for (int i = 0; i < cities->len - 1; i++)
  {
    element = get_element(*cities, i);
    add_element(res_, element);
    buf = copy_array(*cities);
    del_element(&buf, i);
    get_routes(&buf, res_, result, len);
    del_element(res_, res_->len - 1);
  }
}
```

В листинге 3.3 представлена реализация муравьиного алгоритма.

Листинг 3.3 – Реализация муравьиного алгоритма

```
sarray ant_alg(int matrix[N][N], int len, sarray cities, int tmax,
float p, float alpha, float beta)
{
  int q = calc_q(matrix, len);
  sarray best_way = copy_array(cities);
  add_element(&best_way, get_element(best_way, 0));

  int best_cost = get_cost(best_way, matrix);
  int cur_cost = 0;
  float matrix_p[N][N];

  fill_matrix(matrix_p, len, MINPHENOM);
  sant ants[MAXCOUNTOFANT];

  for (int t = 0; t < tmax; t++)
  {
    generate_ants_array(ants, len);
    for (int i = 0; i < len - 1; i++)
    {
      ants_choose_way(ants, matrix_p, matrix, len, alpha, beta);
    }
```

```
22
23      for (int i = 0; i < len; i++)
24      {
25        add_element(&ants[i].way, get_element(ants[i].way, 0));
26      }
27
28      for (int i = 0; i < len; i++)
29      {
30        cur_cost = get_cost(ants[i].way, matrix);
31
32        if (cur_cost < best_cost)
33        {
34          best_cost = cur_cost;
35
36          best_way = copy_array(ants[i].way);
37        }
38      }
39
40      delete_phenom(matrix_p, len, p);
41      add_phenom(matrix, matrix_p, len, q, ants);
42      phenom_correct(matrix_p, len);
43    }
44    return best_way;
45  }
```

В листинге 3.4 представлена реализация алгоритма выбора оптимального пути для муравья.

Листинг 3.4 – Реализация алгоритма выбора оптимального пути для муравья

```
1  void next_city(sant *ants, float matrix_p[N][N],
2  int matrix[N][N], float alpha, float beta)
3  {
4    float number = 0;
5    float denominator = 0;
6    float tao, rev_cost;
7    int cost;
8    int cur_city = ants->way.array[ants->way.len - 1];
9
10   for (int i = 0; i < ants->quere.len; i++)
11   {
12     cost = matrix[cur_city][ants->quere.array[i]];
```

```
13      tao = matrix_p[cur_city][ants->quere.array[i]];
14
15      if (!cost)
16        continue;
17
18      rev_cost = 1.0 / cost;
19      denominator += powf(tao, alpha) + powf(rev_cost, beta);
20    }
21
22    float p_array[N] = {0};
23    float sum = 0;
24
25    for (int i = 0; i < ants->quere.len; i++)
26    {
27      cost = matrix[cur_city][ants->quere.array[i]];
28      tao = matrix_p[cur_city][ants->quere.array[i]];
29
30      rev_cost = 1.0 / cost;
31
32      p_array[i] = (powf(tao, alpha) + powf(rev_cost, beta)) /
          denominator;
33    }
34
35    float x = (float)rand() / RAND_MAX;
36    int index = 0;
37
38    while (x >= 0)
39    {
40      x -= p_array[index];
41      index++;
42    }
43
44    add_element(&ants->way, get_element(ants->quere, index - 1));
45    del_element(&ants->quere, index - 1);
46  }
```

В листинге 3.5 представлена реализация вспомогательных функций.

Листинг 3.5 – Вспомогательные функции

```
1  void add_phenom(int matrix[N][N], float matrix_p[N][N], int len,
2
3  int q, sant ants[MAXCOUNTOFANT])
```

```
4  {
5    int fcity, scity;
6    int cur_cost;
7    float dtao = 0;
8
9    for (int i = 0; i < len; i++)
10   {
11     cur_cost = get_cost(ants[i].way, matrix);
12     dtao += (float)q / cur_cost;
13   }
14
15   for (int i = 0; i < len; i++)
16   {
17     for (int j = 0; j < ants[i].way.len - 1; j++)
18     {
19       fcity = ants[i].way.array[j];
20       scity = ants[i].way.array[j + 1];
21
22       matrix_p[fcity][scity] = matrix_p[fcity][scity] + dtao;
23       matrix_p[scity][fcity] = matrix_p[scity][fcity] + dtao;
24     }
25   }
26 }
27
28 void delete_phenom(float matrix_p[N][N], int len, float p)
29
30 {
31   float qq = 1 - p;
32
33   for (int i = 0; i < len; i++)
34   {
35     for (int j = 0; j < i; j++)
36     {
37       matrix_p[i][j] = matrix_p[i][j] * qq;
38       matrix_p[j][i] = matrix_p[j][i] * qq;
39     }
40   }
41 }
42
43 void phenom_correct(float matrix_p[N][N], int len)
44
```

```
45 {
46   for (int i = 0; i < len; i++)
47   {
48     for (int j = 0; j < i; j++)
49     {
50       if (matrix_p[i][j] <= 0.1)
51       {
52           matrix_p[i][j] = matrix_p[j][i] = 0.1;
53       }
54     }
55   }
56 }
```

3.4 Тестирование

В таблице 3.1 приведены тесты для функции, реализующей алгоритм для решения задачи коммивояжера (алгоритм полного перебора и муравьиный алгоритм). Тесты пройдены успешно.

При тестировании муравьиного алгоритма тесты прошли успешно, так как матрицы смежности маленькие, и поэтому муравьиный алгоритм работает без видимых ошибок, то есть выдает такие же результаты как и алгоритм полного перебора.

Ошибки в результатах муравьиного алгоритма чаще всего возникают при больших размерах матриц (начиная от 9x9). Погрешности бывают чаще всего как разность между наименьшим путем и предпоследним (тот который мог бы быть наименьшим).

Таблица 3.1 – Тестирование функций

Матрица смежности	Ожидаемый результат	Действительный результат
$\begin{pmatrix} 0 & 1 & 10 & 7 \\ 1 & 0 & 1 & 2 \\ 10 & 1 & 0 & 1 \\ 7 & 2 & 1 & 0 \end{pmatrix}$	10, [0, 1, 2, 3, 0]	10, [0, 1, 2, 3, 0]
$\begin{pmatrix} 0 & 3 & 5 & 7 \\ 7 & 0 & 1 & 2 \\ 5 & 1 & 0 & 1 \\ 7 & 2 & 1 & 0 \end{pmatrix}$	11, [2, 3, 1, 0, 2]	11, [2, 3, 1, 0, 2]
$\begin{pmatrix} 0 & 3 & 5 \\ 3 & 0 & 1 \\ 5 & 1 & 0 \end{pmatrix}$	9, [0, 1, 2, 0]	9, [0, 1, 2, 0]

3.5 Вывод

В данном разделе были представлены реализации алгоритма полного перебора и муравьиного алгоритма, а также протестированы.

4 Исследовательская часть

В данном разделе будет произведено сравнение вышеизложенного алгоритма (однопоточная и многопоточная реализация).

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись эксперименты:

- операционная система: Ubuntu 20.04.3 [6] Linux [7] x86_64;
- память: 8 ГБ;
- процессор: 11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz [8].

Эксперименты проводились на ноутбуке, включенном в сеть электропитания. Во время экспериментов ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

4.2 Пример выполнения программы

На рисунках 4.1 4.2 представлен результат работы программы.

В начале выводится матрица смежности, которая поступила на вход.

Далее печатаются результаты работы алгоритма полного перебора и муравьиного алгоритма.

На изображениях графа представлены результаты работы в виде путей. Зеленым отмечен результат работы алгоритма полного перебора, красным – муравьиный алгоритм.

Параметры для муравьиного алгоритма взяты из раздела 4.5 данной главы.

```
Название файла: t.txt
        |0      1       2       3       4       5
-------------------------------------------------------
0       |0      1       10      7       5       15
1       |1      0       1       2       11      10
2       |10     1       0       1       18      2
3       |7      2       1       0       1       12
4       |5      11      18      1       0       2
5       |15     10      2       12      2       0

Результат полного перебора
count: 7
0  1  3  2  5  4  0

Минимальный путь = 13
Время выполнения 16538470

Результат муравьиного алгоритма (последняя итерация)
count: 7
3  2  5  4  0  1  3

Минимальный путь = 13
Время выполнения 5693065.000000
DONE
```

Рисунок 4.1 – Пример работы программы

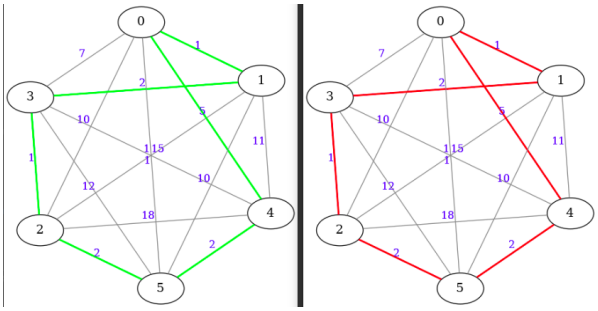

Рисунок 4.2 – Пример работы программы

4.3 Время выполнения алгоритмов

Для проведения временного анализа, программа запускалась на различных размерах графов.

Результаты сравнения времени работы алгоритмов полного перебора и муравьиного алгоритма, приведены в таблице 4.1.

Параметры для муравьиного алгоритма взяты из раздела 4.5 данной главы.

Таблица 4.1 – Время (такт) выполнения полного перебора и муравьиного алгоритма

Размер	**Тип алгоритма**	
	Полный перебор	**Муравьиный**
4	1005398	1000882
5	1145161	1087988
6	1395963	1320744
7	3142151	1574520
8	4743608	1961224
9	42384948	10383369
10	162236576	20311214

На рисунках 4.3 представлен график зависимости времени от размера матрицы смежности для алгоритма полного перебора и муравьиного алгоритма.

Параметры для муравьиного алгоритма взяты из раздела 4.5 данной главы.

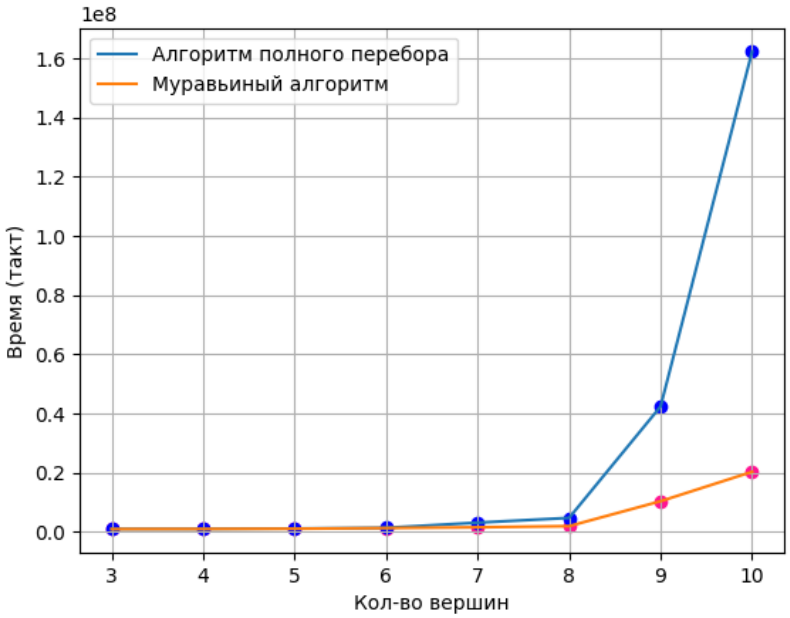

Рисунок 4.3 – Замеры времени работы

4.4 Постановка эксперимента

В муравьином алгоритме вычисления производятся на основе настраиваемых параметров. Рассмотрим три класса данных и подберем к ним параметры, при которых метод даст точный результат при минимальном количестве итераций (итерациями считаются кол-во дней).

Будем рассматривать матрицы размерности 10×10, так как иначе получить точный результат будет довольно просто.

В качестве первого класса данных будет матрица смежности, в которой все значения незначительно отличаются друг от друга, например, в диапазоне $[1, 10]$.

Вторым классом будет матрица, где значения могут отличаться на большей значение, например $[3000, 8000]$.

Третьим классом будет матрица, где значения почти все одинаковые, кроме пару путей, то есть самый короткий путь определен однозначно.

Будем запускать муравьиный алгоритм для всех значений $\alpha, \rho \in [0, 1]$, с шагом $= 0.1$.

В результате тестирования будет выведена таблица со значениями α, ρ, $days$, ΔL_1, ΔL_2, ΔL_3, где

- $days$ — количество дней, данных для решения задачи;
- ΔL_1 - разность эталонного решения задачи и решения муравьиного алгоритма с данными параметрами на первом классе данных.
- ΔL_2 - разность эталонного решения задачи и решения муравьиного алгоритма с данными параметрами на втором классе данных.
- ΔL_3 - разность эталонного решения задачи и решения муравьиного алгоритма с данными параметрами на третьем классе данных.
- α, β, ρ — настроечные параметры.

Класс данных 1

Матрица смежности для данного класса данных (4.1):

$$M = \begin{pmatrix} 0 & 1 & 6 & 7 & 5 & 4 & 4 & 2 & 1 & 2 \\ 1 & 0 & 1 & 2 & 1 & 3 & 2 & 1 & 2 & 1 \\ 6 & 1 & 0 & 1 & 2 & 8 & 2 & 1 & 6 & 5 \\ 7 & 2 & 1 & 0 & 1 & 3 & 1 & 4 & 1 & 8 \\ 5 & 1 & 2 & 1 & 0 & 2 & 2 & 6 & 1 & 7 \\ 4 & 3 & 8 & 3 & 2 & 0 & 3 & 8 & 1 & 9 \\ 4 & 2 & 2 & 1 & 2 & 3 & 0 & 1 & 8 & 2 \\ 2 & 1 & 1 & 4 & 6 & 8 & 1 & 0 & 7 & 3 \\ 1 & 2 & 6 & 1 & 1 & 1 & 8 & 7 & 0 & 1 \\ 2 & 1 & 5 & 8 & 7 & 9 & 2 & 3 & 1 & 0 \end{pmatrix} \tag{4.1}$$

Класс данных 2

Матрица смежности для данного класса данных (4.2):

$$M = \begin{pmatrix} 0 & 3000 & 4599 & 7000 & 3412 & 3388 & 4454 & 3498 & 3541 & 3498 \\ 3000 & 0 & 3607 & 3000 & 3432 & 3033 & 2339 & 5621 & 3984 & 3121 \\ 4599 & 3607 & 0 & 3490 & 3988 & 3834 & 3121 & 8731 & 4454 & 3245 \\ 7000 & 3000 & 3490 & 0 & 3890 & 3444 & 3245 & 3341 & 2339 & 5621 \\ 3412 & 3432 & 3988 & 3890 & 0 & 3004 & 3322 & 7863 & 3121 & 8731 \\ 3388 & 3033 & 3834 & 3444 & 3004 & 0 & 3238 & 3468 & 3245 & 3341 \\ 4454 & 2339 & 3121 & 3245 & 3322 & 3238 & 0 & 3218 & 3322 & 7863 \\ 3498 & 5621 & 8731 & 3341 & 7863 & 3468 & 3218 & 0 & 3238 & 3468 \\ 3541 & 3984 & 4454 & 2339 & 3121 & 3245 & 3322 & 3238 & 0 & 3218 \\ 3498 & 3121 & 3245 & 5621 & 8731 & 3341 & 7863 & 3468 & 3218 & 0 \end{pmatrix} \tag{4.2}$$

Класс данных 3

Матрица смежности для данного класса данных (4.3):

$$M = \begin{pmatrix} 0 & 111 & 111 & 111 & 111 & 111 & 111 & 112 & 111 & 111 \\ 111 & 0 & 112 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 112 & 0 & 111 & 111 & 112 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 0 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 0 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 112 & 111 & 111 & 0 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 0 & 111 & 111 & 111 \\ 112 & 111 & 111 & 111 & 111 & 111 & 111 & 0 & 112 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 112 & 0 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 0 \end{pmatrix} \tag{4.3}$$

Полные результаты данного эксперимента приведены в приложении.

4.5 Вывод

В результате работы в данном разделе можно сделать вывод о том, что сравнительный анализ времени работы двух реализованных алгоритмов указал на факт того, что на всех рассмотренных значениях размерности матрицы смежности муравьиный алгоритм работает лучше, чем алгоритм полного перебора.

Из таблицы 4.1 можно увидеть, что на минимальной рассмотренной размерности матрицы (от 3 до 7) муравьиный алгоритм работает быстрее алгоритма полного перебора на $\approx 5-8\%$. На конечных значениях (от 8 до 10) разрыв увеличивается, и муравьиный алгоритм работает на $\approx 50-60\%$ быстрее.

Из графика, предоставленного на рисунке 4.3, видно, что муравьиный алгоритм до размерности матрицы смежности, равной 7, сравним по скорости работы алгоритма полного перебора. Видное преимущество муравьиного алгоритма будет заметно, начиная с размерности матрицы смежности, равной 8.

На основе проведенной параметризации для двух классов данных можно сделать следующие выводы.

1. Для класса данных, предоставленного в уравнении 4.1 в качестве матрицы смежности, наилучшими наборами стали ($\alpha = 0.6, 0.8, \rho = 0.7, 0.1, 0.4$), так как они показали наиболее стабильные результаты, то $\Delta L_1 = 0$.

2. Для класса данных, предоставленного в уравнении 4.2 в качестве матрицы смежности, наилучшими наборами стали ($\alpha = 0.6, 0.7, \rho = 0.7, 0.3$). При этих параметрах $\Delta L_2 = 0$.

3. Для класса данных, предоставленного в уравнении 4.3 в качестве матрицы смежности, наилучшими наборами стали ($\alpha = 0.5, 0.6, \rho = 0.5, 0.7$). При этих параметрах $\Delta L_3 = 0$.

Из результатов выше можно сделать вывод о том, что наилучшим набором для всех трех классов данных является набор ($\alpha = 0.6, \rho = 0.7$), при этом $tmax$ может равняться любому значению.

Также следует отметить, что из представленных таблицы и графике можно сделать вывод, что трудоемкость алгоритма полного перебора равна $O(n!)$, а муравьиного алгоритма равна $O(tmax * n^3)$ [3].

Заключение

В данной лабораторной работе были рассмотрены основополагающие материалы которые в дальнейшем потребовались при реализации конвейера.

В ходе выполнения лабораторной работы были решены следующие задачи.

1. Изучены основы алгоритма полного перебора.
2. Применены изученные основы алгоритма полного перебора для реализации.
3. Изучены основы муравьиного алгоритма.
4. Применены изученные основы муравьиного алгоритма для реализации.
5. Проведена параметризация муравьиного алгоритма на трех классах данных.
6. Проведен сравнительный анализ времени работы реализованных алгоритмов.
7. Подготовлен отчет о лабораторной работе.

Поставленная цель достигнута.

Литература

[1] Задача коммирояжера [Электронный ресурс]. Режим доступа: http://galyautdinov.ru/post/zadacha-kommivoyazhera (дата обращения: 30.10.2021).

[2] Гамильтонов цикл [Электронный ресурс]. Режим доступа: https://bigenc.ru/mathematics/text/2343214 (дата обращения: 30.10.2021).

[3] М.В. Ульянов. Ресурсно-эффективные компьютерные алгоритмы. ФИЗМАТЛИТ, 2008. с. 304.

[4] Введение в язык Си [Электронный ресурс]. Режим доступа: http://dfe.petrsu.ru/koi/posob/c/ (дата обращения: 02.11.2021).

[5] Visual Studio Code [Электронный ресурс]. Режим доступа: https://code.visualstudio.com/ (дата обращения: 30.09.2021).

[6] Ubuntu 20.04.3 LTS (Focal Fossa) [Электронный ресурс]. Режим доступа: https://releases.ubuntu.com/20.04/ (дата обращения: 04.09.2021).

[7] Linux – Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Linux (дата обращения: 04.09.2021).

[8] Процессор Intel® Core™ i5-1135G7 [Электронный ресурс]. Режим доступа: https://www.intel.ru/content/www/ru/ru/products/sku/208658/intel-core-i51135g7-processor-8m-cache-up-to-4-20-ghz/specifications.html (дата обращения: 04.09.2021).

Приложение

Таблица 4.2 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.1	0.1	150	0	816	1
0.1	0.1	200	1	1245	1
0.1	0.1	250	5	0	1
0.1	0.1	300	0	816	0
0.1	0.1	350	1	816	0
0.1	0.1	400	5	0	0
0.1	0.1	450	1	816	0
0.1	0.1	500	0	0	0
0.2	0.1	100	11	816	1
0.2	0.1	150	1	816	1
0.2	0.1	200	1	0	0
0.2	0.1	250	0	0	0
0.2	0.1	300	0	0	1
0.2	0.1	350	0	0	0
0.2	0.1	400	0	0	1
0.2	0.1	450	0	0	0
0.2	0.1	500	0	0	0
0.3	0.1	100	1	0	1
0.3	0.1	150	5	816	0
0.3	0.1	200	5	0	0
0.3	0.1	250	0	0	1
0.3	0.1	300	0	816	0
0.3	0.1	350	0	0	0
0.3	0.1	400	0	0	0
0.3	0.1	450	0	0	0
0.3	0.1	500	0	0	0
0.4	0.1	100	1	0	1
0.4	0.1	150	0	894	0
0.4	0.1	200	0	0	0
0.4	0.1	250	0	816	0
0.4	0.1	300	0	0	0
0.4	0.1	350	0	0	0
0.4	0.1	400	0	0	0
0.4	0.1	450	0	0	0
0.4	0.1	500	0	0	0

Таблица 4.3 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.5	0.1	150	0	0	1
0.5	0.1	200	5	0	1
0.5	0.1	250	1	894	0
0.5	0.1	300	0	816	0
0.5	0.1	350	0	0	0
0.5	0.1	400	0	0	1
0.5	0.1	450	0	0	0
0.5	0.1	500	0	894	0
0.6	0.1	100	5	0	1
0.6	0.1	150	0	894	0
0.6	0.1	200	1	1245	1
0.6	0.1	250	1	0	0
0.6	0.1	300	0	816	0
0.6	0.1	350	0	816	0
0.6	0.1	400	1	894	0
0.6	0.1	450	0	0	0
0.6	0.1	500	0	0	0
0.7	0.1	100	1	1245	0
0.7	0.1	150	0	1230	0
0.7	0.1	200	5	816	1
0.7	0.1	250	0	816	0
0.7	0.1	300	0	0	1
0.7	0.1	350	0	0	0
0.7	0.1	400	0	0	0
0.7	0.1	450	0	0	0
0.7	0.1	500	0	0	0
0.8	0.1	100	0	0	1
0.8	0.1	150	0	816	0
0.8	0.1	200	0	816	0
0.8	0.1	250	0	894	0
0.8	0.1	300	0	894	0
0.8	0.1	350	0	816	0
0.8	0.1	400	0	0	0
0.8	0.1	450	0	0	0
0.8	0.1	500	0	0	0

Таблица 4.4 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.9	0.1	150	1	0	0
0.9	0.1	200	0	0	1
0.9	0.1	250	1	0	0
0.9	0.1	300	0	894	0
0.9	0.1	350	0	0	0
0.9	0.1	400	0	0	0
0.9	0.1	450	0	0	0
0.9	0.1	500	0	0	0
0.1	0.2	100	7	894	1
0.1	0.2	150	1	0	0
0.1	0.2	200	0	894	1
0.1	0.2	250	1	0	0
0.1	0.2	300	1	0	1
0.1	0.2	350	0	0	0
0.1	0.2	400	0	0	5
0.1	0.2	450	1	0	0
0.1	0.2	500	0	0	0
0.2	0.2	100	1	1292	1
0.2	0.2	150	0	0	1
0.2	0.2	200	0	0	0
0.2	0.2	250	1	0	1
0.2	0.2	300	0	0	0
0.2	0.2	350	0	894	0
0.2	0.2	400	0	0	1
0.2	0.2	450	5	894	0
0.2	0.2	500	0	0	0
0.3	0.2	100	0	0	0
0.3	0.2	150	0	816	0
0.3	0.2	200	0	0	0
0.3	0.2	250	1	0	1
0.3	0.2	300	0	0	1
0.3	0.2	350	0	0	0
0.3	0.2	400	0	0	0
0.3	0.2	450	0	0	1
0.3	0.2	500	0	0	0

Таблица 4.5 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.4	0.2	150	1	816	0
0.4	0.2	200	0	0	0
0.4	0.2	250	0	0	0
0.4	0.2	300	0	0	0
0.4	0.2	350	0	0	0
0.4	0.2	400	0	0	0
0.4	0.2	450	0	0	1
0.4	0.2	500	0	0	0
0.5	0.2	100	6	0	1
0.5	0.2	150	0	816	0
0.5	0.2	200	0	816	0
0.5	0.2	250	0	816	0
0.5	0.2	300	0	894	0
0.5	0.2	350	0	0	0
0.5	0.2	400	0	0	0
0.5	0.2	450	0	0	0
0.5	0.2	500	0	0	0
0.6	0.2	100	0	894	0
0.6	0.2	150	0	0	0
0.6	0.2	200	0	816	0
0.6	0.2	250	0	894	0
0.6	0.2	300	0	816	1
0.6	0.2	350	0	894	0
0.6	0.2	400	0	816	0
0.6	0.2	450	0	0	0
0.6	0.2	500	0	0	0
0.7	0.2	100	0	894	1
0.7	0.2	150	0	894	0
0.7	0.2	200	0	0	0
0.7	0.2	250	0	816	0
0.7	0.2	300	0	816	0
0.7	0.2	350	0	0	0
0.7	0.2	400	0	816	0
0.7	0.2	450	0	0	0
0.7	0.2	500	0	0	0

Таблица 4.6 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.8	0.2	150	0	816	0
0.8	0.2	200	0	0	0
0.8	0.2	250	0	894	0
0.8	0.2	300	5	894	0
0.8	0.2	350	0	0	0
0.8	0.2	400	0	816	0
0.8	0.2	450	0	0	0
0.8	0.2	500	0	0	0
0.9	0.2	100	9	816	1
0.9	0.2	150	1	0	1
0.9	0.2	200	1	816	0
0.9	0.2	250	0	816	0
0.9	0.2	300	0	816	0
0.9	0.2	350	0	0	0
0.9	0.2	400	0	0	0
0.9	0.2	450	0	816	0
0.9	0.2	500	0	0	0
0.1	0.3	100	0	894	1
0.1	0.3	150	1	1245	0
0.1	0.3	200	0	0	0
0.1	0.3	250	0	1245	0
0.1	0.3	300	0	816	0
0.1	0.3	350	1	0	6
0.1	0.3	400	0	0	0
0.1	0.3	450	1	0	0
0.1	0.3	500	1	0	0
0.2	0.3	100	6	894	0
0.2	0.3	150	0	0	1
0.2	0.3	200	0	816	1
0.2	0.3	250	0	0	1
0.2	0.3	300	0	816	1
0.2	0.3	350	0	894	1
0.2	0.3	400	0	0	0
0.2	0.3	450	0	0	0
0.2	0.3	500	0	0	0

Таблица 4.7 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.3	0.3	150	5	1245	1
0.3	0.3	200	0	0	0
0.3	0.3	250	0	0	0
0.3	0.3	300	0	0	0
0.3	0.3	350	0	0	0
0.3	0.3	400	0	1230	0
0.3	0.3	450	0	0	0
0.3	0.3	500	0	0	0
0.4	0.3	100	5	0	0
0.4	0.3	150	0	0	0
0.4	0.3	200	0	816	0
0.4	0.3	250	0	0	0
0.4	0.3	300	0	816	0
0.4	0.3	350	0	1230	0
0.4	0.3	400	0	0	0
0.4	0.3	450	0	0	0
0.4	0.3	500	0	0	0
0.5	0.3	100	5	894	0
0.5	0.3	150	0	0	0
0.5	0.3	200	5	894	0
0.5	0.3	250	0	816	1
0.5	0.3	300	0	816	0
0.5	0.3	350	0	816	0
0.5	0.3	400	0	0	0
0.5	0.3	450	0	0	0
0.5	0.3	500	0	0	0
0.6	0.3	100	0	1619	0
0.6	0.3	150	5	1292	1
0.6	0.3	200	0	816	0
0.6	0.3	250	0	0	0
0.6	0.3	300	0	0	0
0.6	0.3	350	0	894	0
0.6	0.3	400	0	0	0
0.6	0.3	450	0	894	0
0.6	0.3	500	0	894	0

Таблица 4.8 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.7	0.3	150	0	1391	0
0.7	0.3	200	0	0	1
0.7	0.3	250	0	0	0
0.7	0.3	300	1	816	0
0.7	0.3	350	0	0	0
0.7	0.3	400	0	0	0
0.7	0.3	450	0	0	0
0.7	0.3	500	0	0	0
0.8	0.3	100	0	0	1
0.8	0.3	150	0	894	1
0.8	0.3	200	0	1230	0
0.8	0.3	250	0	894	0
0.8	0.3	300	0	0	0
0.8	0.3	350	0	0	0
0.8	0.3	400	0	0	0
0.8	0.3	450	0	816	0
0.8	0.3	500	0	0	0
0.9	0.3	100	0	894	0
0.9	0.3	150	0	0	0
0.9	0.3	200	0	816	0
0.9	0.3	250	0	816	0
0.9	0.3	300	0	0	0
0.9	0.3	350	0	0	0
0.9	0.3	400	0	0	0
0.9	0.3	450	0	0	0
0.9	0.3	500	0	0	0
0.1	0.4	100	5	0	0
0.1	0.4	150	7	0	9
0.1	0.4	200	1	0	0
0.1	0.4	250	0	816	1
0.1	0.4	300	0	0	1
0.1	0.4	350	0	0	0
0.1	0.4	400	0	816	0
0.1	0.4	450	0	0	0
0.1	0.4	500	0	0	0

Таблица 4.9 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.2	0.4	150	6	1230	0
0.2	0.4	200	0	0	0
0.2	0.4	250	0	0	0
0.2	0.4	300	0	816	0
0.2	0.4	350	0	816	1
0.2	0.4	400	0	0	0
0.2	0.4	450	1	0	0
0.2	0.4	500	0	0	0
0.3	0.4	100	1	0	0
0.3	0.4	150	0	1230	0
0.3	0.4	200	0	0	0
0.3	0.4	250	0	0	0
0.3	0.4	300	0	816	0
0.3	0.4	350	0	816	1
0.3	0.4	400	1	0	0
0.3	0.4	450	0	0	0
0.3	0.4	500	0	0	0
0.4	0.4	100	1	816	1
0.4	0.4	150	0	894	0
0.4	0.4	200	0	894	0
0.4	0.4	250	0	0	0
0.4	0.4	300	0	0	1
0.4	0.4	350	0	0	0
0.4	0.4	400	0	816	0
0.4	0.4	450	0	0	0
0.4	0.4	500	0	0	0
0.5	0.4	100	0	816	1
0.5	0.4	150	0	0	0
0.5	0.4	200	1	0	1
0.5	0.4	250	0	0	0
0.5	0.4	300	0	0	0
0.5	0.4	350	0	894	0
0.5	0.4	400	0	0	0
0.5	0.4	450	0	0	0
0.5	0.4	500	0	0	0

Таблица 4.10 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.6	0.4	150	0	816	0
0.6	0.4	200	0	0	0
0.6	0.4	250	0	0	0
0.6	0.4	300	0	0	0
0.6	0.4	350	0	816	0
0.6	0.4	400	0	0	0
0.6	0.4	450	0	0	0
0.6	0.4	500	0	0	0
0.7	0.4	100	1	0	0
0.7	0.4	150	5	0	0
0.7	0.4	200	0	0	0
0.7	0.4	250	0	0	0
0.7	0.4	300	0	816	0
0.7	0.4	350	0	0	0
0.7	0.4	400	0	0	0
0.7	0.4	450	0	0	0
0.7	0.4	500	0	816	0
0.8	0.4	100	1	1292	1
0.8	0.4	150	0	1245	0
0.8	0.4	200	0	816	0
0.8	0.4	250	0	816	1
0.8	0.4	300	0	0	0
0.8	0.4	350	0	0	0
0.8	0.4	400	0	894	0
0.8	0.4	450	0	0	0
0.8	0.4	500	0	0	0
0.9	0.4	100	6	1230	1
0.9	0.4	150	0	0	1
0.9	0.4	200	0	816	0
0.9	0.4	250	0	0	0
0.9	0.4	300	0	0	0
0.9	0.4	350	0	0	0
0.9	0.4	400	0	816	0
0.9	0.4	450	0	0	0
0.9	0.4	500	0	0	0

Таблица 4.11 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.1	0.5	150	0	816	0
0.1	0.5	200	1	816	0
0.1	0.5	250	1	0	0
0.1	0.5	300	0	0	0
0.1	0.5	350	1	0	0
0.1	0.5	400	7	816	0
0.1	0.5	450	0	0	1
0.1	0.5	500	0	0	0
0.2	0.5	100	0	0	0
0.2	0.5	150	1	816	0
0.2	0.5	200	0	816	1
0.2	0.5	250	0	816	0
0.2	0.5	300	0	0	0
0.2	0.5	350	0	0	0
0.2	0.5	400	0	816	0
0.2	0.5	450	0	0	0
0.2	0.5	500	0	0	0
0.3	0.5	100	1	894	0
0.3	0.5	150	6	0	0
0.3	0.5	200	1	0	0
0.3	0.5	250	1	0	0
0.3	0.5	300	0	0	0
0.3	0.5	350	0	0	0
0.3	0.5	400	0	816	0
0.3	0.5	450	0	0	0
0.3	0.5	500	0	0	0
0.4	0.5	100	0	0	0
0.4	0.5	150	6	816	0
0.4	0.5	200	0	1245	0
0.4	0.5	250	0	816	0
0.4	0.5	300	0	0	0
0.4	0.5	350	0	0	0
0.4	0.5	400	5	0	0
0.4	0.5	450	0	0	0
0.4	0.5	500	0	816	0

Таблица 4.12 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.5	0.5	150	0	816	0
0.5	0.5	200	0	816	0
0.5	0.5	250	1	894	0
0.5	0.5	300	0	0	0
0.5	0.5	350	0	0	0
0.5	0.5	400	0	0	0
0.5	0.5	450	0	0	0
0.5	0.5	500	0	0	0
0.6	0.5	100	0	1778	0
0.6	0.5	150	0	0	0
0.6	0.5	200	0	894	0
0.6	0.5	250	1	816	0
0.6	0.5	300	0	0	0
0.6	0.5	350	0	894	0
0.6	0.5	400	0	816	0
0.6	0.5	450	0	0	0
0.6	0.5	500	0	0	0
0.7	0.5	100	0	1292	0
0.7	0.5	150	0	816	0
0.7	0.5	200	0	0	0
0.7	0.5	250	0	0	0
0.7	0.5	300	0	0	0
0.7	0.5	350	0	0	0
0.7	0.5	400	0	0	0
0.7	0.5	450	0	816	0
0.7	0.5	500	0	0	0
0.8	0.5	100	1	1549	0
0.8	0.5	150	0	1245	0
0.8	0.5	200	0	816	0
0.8	0.5	250	1	0	0
0.8	0.5	300	0	0	0
0.8	0.5	350	1	0	0
0.8	0.5	400	0	0	0
0.8	0.5	450	0	0	0
0.8	0.5	500	0	816	0

Таблица 4.13 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.9	0.5	150	0	1292	0
0.9	0.5	200	0	1391	0
0.9	0.5	250	0	0	0
0.9	0.5	300	0	816	0
0.9	0.5	350	0	0	0
0.9	0.5	400	0	0	0
0.9	0.5	450	0	0	0
0.9	0.5	500	0	0	0
0.1	0.6	100	0	0	0
0.1	0.6	150	1	1230	0
0.1	0.6	200	1	0	0
0.1	0.6	250	1	0	0
0.1	0.6	300	0	0	0
0.1	0.6	350	0	816	0
0.1	0.6	400	0	0	0
0.1	0.6	450	0	816	0
0.1	0.6	500	0	0	0
0.2	0.6	100	6	816	5
0.2	0.6	150	0	816	0
0.2	0.6	200	0	0	0
0.2	0.6	250	1	0	0
0.2	0.6	300	0	0	0
0.2	0.6	350	0	816	0
0.2	0.6	400	1	0	0
0.2	0.6	450	0	816	0
0.2	0.6	500	0	816	0
0.3	0.6	100	0	0	1
0.3	0.6	150	0	1230	0
0.3	0.6	200	0	816	0
0.3	0.6	250	0	0	1
0.3	0.6	300	0	0	0
0.3	0.6	350	0	0	0
0.3	0.6	400	0	894	0
0.3	0.6	450	0	0	0
0.3	0.6	500	0	894	0

Таблица 4.14 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.4	0.6	150	5	894	0
0.4	0.6	200	0	1292	1
0.4	0.6	250	1	0	0
0.4	0.6	300	0	816	0
0.4	0.6	350	0	0	0
0.4	0.6	400	0	0	0
0.4	0.6	450	0	0	0
0.4	0.6	500	0	0	0
0.5	0.6	100	0	816	0
0.5	0.6	150	1	0	0
0.5	0.6	200	0	816	1
0.5	0.6	250	0	0	0
0.5	0.6	300	0	0	0
0.5	0.6	350	1	0	0
0.5	0.6	400	0	0	0
0.5	0.6	450	0	0	0
0.5	0.6	500	0	0	0
0.6	0.6	100	0	816	1
0.6	0.6	150	0	1230	6
0.6	0.6	200	0	0	0
0.6	0.6	250	0	816	0
0.6	0.6	300	0	0	0
0.6	0.6	350	0	0	0
0.6	0.6	400	0	0	0
0.6	0.6	450	0	816	0
0.6	0.6	500	0	816	0
0.7	0.6	100	1	894	1
0.7	0.6	150	0	1245	0
0.7	0.6	200	0	894	0
0.7	0.6	250	0	816	0
0.7	0.6	300	0	816	0
0.7	0.6	350	0	816	0
0.7	0.6	400	0	0	0
0.7	0.6	450	0	0	0
0.7	0.6	500	0	0	0

Таблица 4.15 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.8	0.6	150	0	816	0
0.8	0.6	200	0	816	0
0.8	0.6	250	0	894	0
0.8	0.6	300	0	816	0
0.8	0.6	350	0	0	0
0.8	0.6	400	0	0	0
0.8	0.6	450	0	0	1
0.8	0.6	500	0	0	0
0.9	0.6	100	1	1391	0
0.9	0.6	150	0	1785	1
0.9	0.6	200	0	816	0
0.9	0.6	250	0	816	0
0.9	0.6	300	0	0	0
0.9	0.6	350	0	0	0
0.9	0.6	400	0	816	0
0.9	0.6	450	0	0	0
0.9	0.6	500	0	0	0
0.1	0.7	100	0	894	6
0.1	0.7	150	0	816	0
0.1	0.7	200	0	0	0
0.1	0.7	250	0	0	1
0.1	0.7	300	0	0	0
0.1	0.7	350	0	0	1
0.1	0.7	400	0	816	0
0.1	0.7	450	0	0	0
0.1	0.7	500	1	0	0
0.2	0.7	100	1	816	1
0.2	0.7	150	0	0	1
0.2	0.7	200	1	0	1
0.2	0.7	250	0	0	0
0.2	0.7	300	0	894	1
0.2	0.7	350	0	0	0
0.2	0.7	400	0	0	0
0.2	0.7	450	0	816	0
0.2	0.7	500	0	0	0

Таблица 4.16 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.3	0.7	150	1	0	1
0.3	0.7	200	0	0	0
0.3	0.7	250	1	1230	0
0.3	0.7	300	1	816	0
0.3	0.7	350	0	0	0
0.3	0.7	400	0	0	0
0.3	0.7	450	0	0	0
0.3	0.7	500	0	0	0
0.4	0.7	100	0	816	0
0.4	0.7	150	0	816	1
0.4	0.7	200	0	816	1
0.4	0.7	250	0	1230	0
0.4	0.7	300	0	0	0
0.4	0.7	350	0	816	1
0.4	0.7	400	0	816	0
0.4	0.7	450	0	0	0
0.4	0.7	500	0	0	0
0.5	0.7	100	9	816	1
0.5	0.7	150	1	1245	0
0.5	0.7	200	0	816	0
0.5	0.7	250	0	0	0
0.5	0.7	300	0	0	0
0.5	0.7	350	0	0	0
0.5	0.7	400	0	816	0
0.5	0.7	450	0	0	0
0.5	0.7	500	0	816	0
0.6	0.7	100	0	816	1
0.6	0.7	150	0	1245	1
0.6	0.7	200	0	0	0
0.6	0.7	250	0	0	0
0.6	0.7	300	0	0	0
0.6	0.7	350	0	0	0
0.6	0.7	400	0	0	0
0.6	0.7	450	0	0	0
0.6	0.7	500	0	0	0

Таблица 4.17 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.7	0.7	150	0	0	1
0.7	0.7	200	1	0	0
0.7	0.7	250	0	0	0
0.7	0.7	300	0	816	0
0.7	0.7	350	1	0	0
0.7	0.7	400	0	0	0
0.7	0.7	450	0	0	0
0.7	0.7	500	0	0	0
0.8	0.7	100	0	1292	0
0.8	0.7	150	0	0	1
0.8	0.7	200	0	816	0
0.8	0.7	250	0	816	0
0.8	0.7	300	0	0	0
0.8	0.7	350	0	0	0
0.8	0.7	400	0	0	0
0.8	0.7	450	0	0	0
0.8	0.7	500	0	816	0
0.9	0.7	100	0	1391	0
0.9	0.7	150	0	0	0
0.9	0.7	200	0	0	0
0.9	0.7	250	0	894	1
0.9	0.7	300	0	0	0
0.9	0.7	350	0	816	0
0.9	0.7	400	0	0	0
0.9	0.7	450	0	0	0
0.9	0.7	500	0	816	0
0.1	0.8	100	1	894	0
0.1	0.8	150	0	0	1
0.1	0.8	200	0	0	1
0.1	0.8	250	0	816	0
0.1	0.8	300	0	816	0
0.1	0.8	350	0	0	0
0.1	0.8	400	0	0	0
0.1	0.8	450	5	0	0
0.1	0.8	500	1	816	0

Таблица 4.18 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.2	0.8	150	0	0	1
0.2	0.8	200	0	0	0
0.2	0.8	250	0	894	0
0.2	0.8	300	0	816	1
0.2	0.8	350	0	0	0
0.2	0.8	400	0	0	1
0.2	0.8	450	0	0	0
0.2	0.8	500	0	0	0
0.3	0.8	100	0	1230	0
0.3	0.8	150	0	816	0
0.3	0.8	200	0	894	0
0.3	0.8	250	0	0	0
0.3	0.8	300	1	0	0
0.3	0.8	350	1	0	0
0.3	0.8	400	0	1230	0
0.3	0.8	450	0	0	1
0.3	0.8	500	0	0	0
0.4	0.8	100	0	894	1
0.4	0.8	150	1	816	1
0.4	0.8	200	1	894	0
0.4	0.8	250	1	1230	0
0.4	0.8	300	1	0	0
0.4	0.8	350	0	0	0
0.4	0.8	400	0	0	0
0.4	0.8	450	0	0	0
0.4	0.8	500	0	0	0
0.5	0.8	100	1	816	1
0.5	0.8	150	5	894	1
0.5	0.8	200	0	894	0
0.5	0.8	250	5	894	0
0.5	0.8	300	1	0	0
0.5	0.8	350	0	0	0
0.5	0.8	400	1	894	0
0.5	0.8	450	0	0	0
0.5	0.8	500	0	0	0

Таблица 4.19 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.6	0.8	150	0	0	0
0.6	0.8	200	0	0	0
0.6	0.8	250	0	0	0
0.6	0.8	300	0	0	0
0.6	0.8	350	0	816	0
0.6	0.8	400	1	0	0
0.6	0.8	450	0	0	0
0.6	0.8	500	0	0	0
0.7	0.8	100	0	1391	0
0.7	0.8	150	0	816	1
0.7	0.8	200	0	816	0
0.7	0.8	250	0	0	0
0.7	0.8	300	0	816	0
0.7	0.8	350	0	0	0
0.7	0.8	400	1	0	0
0.7	0.8	450	0	816	0
0.7	0.8	500	0	0	0
0.8	0.8	100	0	0	0
0.8	0.8	150	0	0	0
0.8	0.8	200	1	1245	0
0.8	0.8	250	0	0	0
0.8	0.8	300	0	0	0
0.8	0.8	350	0	0	0
0.8	0.8	400	0	0	0
0.8	0.8	450	0	0	0
0.8	0.8	500	0	816	0
0.9	0.8	100	1	1230	0
0.9	0.8	150	0	1230	0
0.9	0.8	200	0	1245	0
0.9	0.8	250	0	0	0
0.9	0.8	300	0	0	0
0.9	0.8	350	0	816	0
0.9	0.8	400	0	894	0
0.9	0.8	450	0	816	0
0.9	0.8	500	0	0	0

Таблица 4.20 – Время (такт) выполнения полного перебора и муравьиного алгоритма

α	ρ	t_{max}	ΔL_1	ΔL_2	ΔL_3
0.1	0.9	150	6	894	0
0.1	0.9	200	0	0	1
0.1	0.9	250	0	0	0
0.1	0.9	300	0	0	0
0.1	0.9	350	5	816	0
0.1	0.9	400	0	0	0
0.1	0.9	450	0	0	0
0.1	0.9	500	0	0	0
0.2	0.9	100	0	1778	1
0.2	0.9	150	0	1245	0
0.2	0.9	200	1	0	1
0.2	0.9	250	0	816	0
0.2	0.9	300	0	0	0
0.2	0.9	350	0	0	0
0.2	0.9	400	0	0	0
0.2	0.9	450	0	0	0
0.2	0.9	500	0	0	0
0.3	0.9	100	5	1230	0
0.3	0.9	150	0	0	0
0.3	0.9	200	1	1245	0
0.3	0.9	250	0	816	0
0.3	0.9	300	0	816	0
0.3	0.9	350	0	0	0
0.3	0.9	400	0	0	0
0.3	0.9	450	1	816	0
0.3	0.9	500	0	816	0
0.4	0.9	100	5	0	0
0.4	0.9	150	5	816	0
0.4	0.9	200	0	0	1
0.4	0.9	250	0	894	0
0.4	0.9	300	0	0	0
0.4	0.9	350	0	0	0
0.4	0.9	400	0	0	0
0.4	0.9	450	0	816	0
0.4	0.9	500	0	0	0
0.5	0.9	100	1	0	0
0.5	0.9	150	1	894	0