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»

на тему: «Методы решения задачи коммивояжера»

Студент группы ИУ7-54Б _____ (Подпись, дата) Спирин М. П. (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Волкова Л. Л. (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Строганов Ю. В.. (Фамилия И.О.)

Москва — 2023 г.

Содержание

Введение

Задача коммивояжёра – задача транспортной логистики, отрасли, занимающейся планированием транспортных перевозок. Коммивояжёру, чтобы распродать нужные и не очень нужные в хозяйстве товары, следует объехать n пунктов и в конце концов вернуться в исходный пункт. Требуется определить наиболее выгодный маршрут объезда. В качестве меры выгодности маршрута может служить суммарное время в пути, суммарная стоимость дороги, или, в простейшем случае, длина маршрута.

Муравьиный алгоритм – полиномиальный алгоритм 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.

Целью данной лабораторной работы является параметризация метода решения задачи коммивояжера на основе муравьиного метода.

Для поставленной цели необходимо выполнить следующие задачи.

1) Описать задачу коммивояжера.

2) Описать методы решения задачи коммивояжера — метод полного перебора и метод на основе муравьиного алгоритма.

3) Привести схемы муравьиного алгоритма и алгоритма, позволяющего решить задачу коммивояжера методом полного перебора.

4) Разработать и реализовать программный продукт, позволяющий решить задачу коммивояжера исследуемыми методами.

5) Сравнить по времени метод полного перебора и метод на основе муравьиного алгоритма.

6) Описать и обосновать полученные результаты в отчете о выполненной лабораторной работе.

Выданный индивидуальный вариант для выполнения лабораторной работы:

— ориентированый граф;

— с элитными муравьев;

— маршрут без возврата в исходный порт.

1 Аналитическая часть

В этом разделе будет представлена информация о задаче коммивояжера, а также о способах её решения — методе полного перебора и методе на основе муравьиного алгоритма.

Задача коммивояжера (англ. *traveling salesman problem*) — (задача о бродячем торговце) одна из самых важных задач всей транспортной логистики, в которой рассматриваются вершины графа, а также матрица смежности (для расстояния между вершинами) [1]. Задача заключается в том, чтобы найти такой порядок посещения вершин графа, при котором путь будет минимален, каждая вершина будет посещена лишь один раз, а возврат произойдет в начальную вершину.

Полный перебор для задачи коммивояжера имеет высокую сложность алгоритма ($n!$), где n — количество портов [2]. Суть в полном переборе всех возможных путей в графе и выбор наименьшего из них. Решение будет получено, но имеются большие затраты по времени выполнения при уже небольшом количестве вершин в графе.

1.1 Метод на основе муравьиного алгоритма

Муравьиный алгоритм (англ. *ant colony optimization*) — метод решения задачи оптимизации, основанный на принципе поведения колонии муравьев [2].

Муравьи действуют, руководствуясь органами чувств. Каждый муравей оставляет на своем пути феромоны, чтобы другие могли по ним ориентироваться. При большом количестве муравьев наибольшее количество феромона остается на наиболее посещаемом пути, посещаемость же связана с длинами ребер.

Суть в том, что отдельно взятый муравей мало что может, поскольку он способен выполнять только максимально простые задачи. Но при большом числе других таких муравьев они могут выступать самостоятельными вычислительными единицами. Муравьи используют непрямой обмен

информацией через окружающую среду посредством феромона.

Пусть муравей имеет следующие характеристики:

1) зрение — способность определить длину ребра;

2) память — способность запомнить пройденный маршрут;

3) обоняние — способность чуять феромон.

Также введем функцию, характеризующую привлекательность ребра, определяемую благодаря зрению:

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

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

Также понадобится формула вычисления вероятности перехода в заданную точку:

$$p_{k,ij} = \begin{cases} \frac{\eta_{ij}^{\alpha} \cdot \tau_{ij}^{\beta}}{\sum_{q \notin J_k} \eta_{iq}^{\alpha} \cdot \tau_{iq}^{\beta}}, j \notin J_k, \\ 0, j \in J_k, \end{cases} \tag{1.2}$$

где a — параметр влияния длины пути, b — параметр влияния феромона, τ_{ij} — количество феромонов на ребре ij, η_{ij} — привлекательность ребра ij, J_k — список посещенных за текущий день портов.

После завершения движения всех муравьев (ночью, перед наступлением следующего дня), феромон обновляется по следующей формуле:

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

При этом величина, на которую меняется количество феромона, считается по формуле:

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

где

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} Q/L_k, \text{ребро посещено муравьем } k \text{ в текущий день } t, \\ 0, \text{иначе.} \end{cases} \tag{1.5}$$

Q настраивает концентрацию феромонов и должно быть соразмерно длине лучшего найденного пути.

Поскольку вероятность (1.2) перехода в заданную точку не должна быть равна нулю, необходимо обеспечить неравенство $\tau_{ij}(t)$ нулю посредством введения дополнительного минимально возможного значения феромона τ_{min} и в случае, если $\tau_{ij}(t+1)$ принимает значение, меньшее τ_{min}, откатывать феромон до этой величины.

Для выбора пути используется набор ограничений.

1) Каждый муравей имеет список запретов — список уже посещенных портов (вершин графа).

2) Муравьиное зрение отвечает за эвристическое желание посетить вершину.

3) Муравьиное обоняние отвечает за ощущение феромона на определенном пути (ребре). При этом количество феромона на пути (ребре) в день t обозначается как $\tau_{i,j}(t)$.

4) После прохождения определенного ребра муравей откладывает на нем некоторое количество феромона, которое показывает оптимальность сделанного выбора, количество вычисляется по формуле (1.5).

Следует упомянуть, что сумма коэффициентов α и β равна 1.

Элитные муравьи

В данной реализации муравьиного алгоритма используются элитные муравьи — после дня жизни они откладывают феромоны на лучшем пути, словно прошли обычные муравьи.

Вывод

В данном разделе была рассмотрена задача коммивояжера, а также способы её решения — полный перебор и методе на основе муравьиного алгоритма.

2 Конструкторская часть

В данном разделе будут представлены схемы алгоритма полного перебора и муравьиного алгоритма.

2.1 Требования к ПО

К программе предъявлены ряд требований:

— программа должна получать на вход матрицу смежности, для которой можно будет выбрать один из алгоритмов поиска оптимальных путей (полным перебором или муравьиным алгоритмом);

— программа должна позволять пользователю определять коэффициенты и количество дней для метода на основе муравьиного алгоритма;

— программа должна давать возможность получить минимальную сумму пути, а также сам путь, используя один из алгоритмов.

2.2 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма полного перебора путей. На рисунках 2.2 и 2.3 представлена схема муравьиного алгоритма.

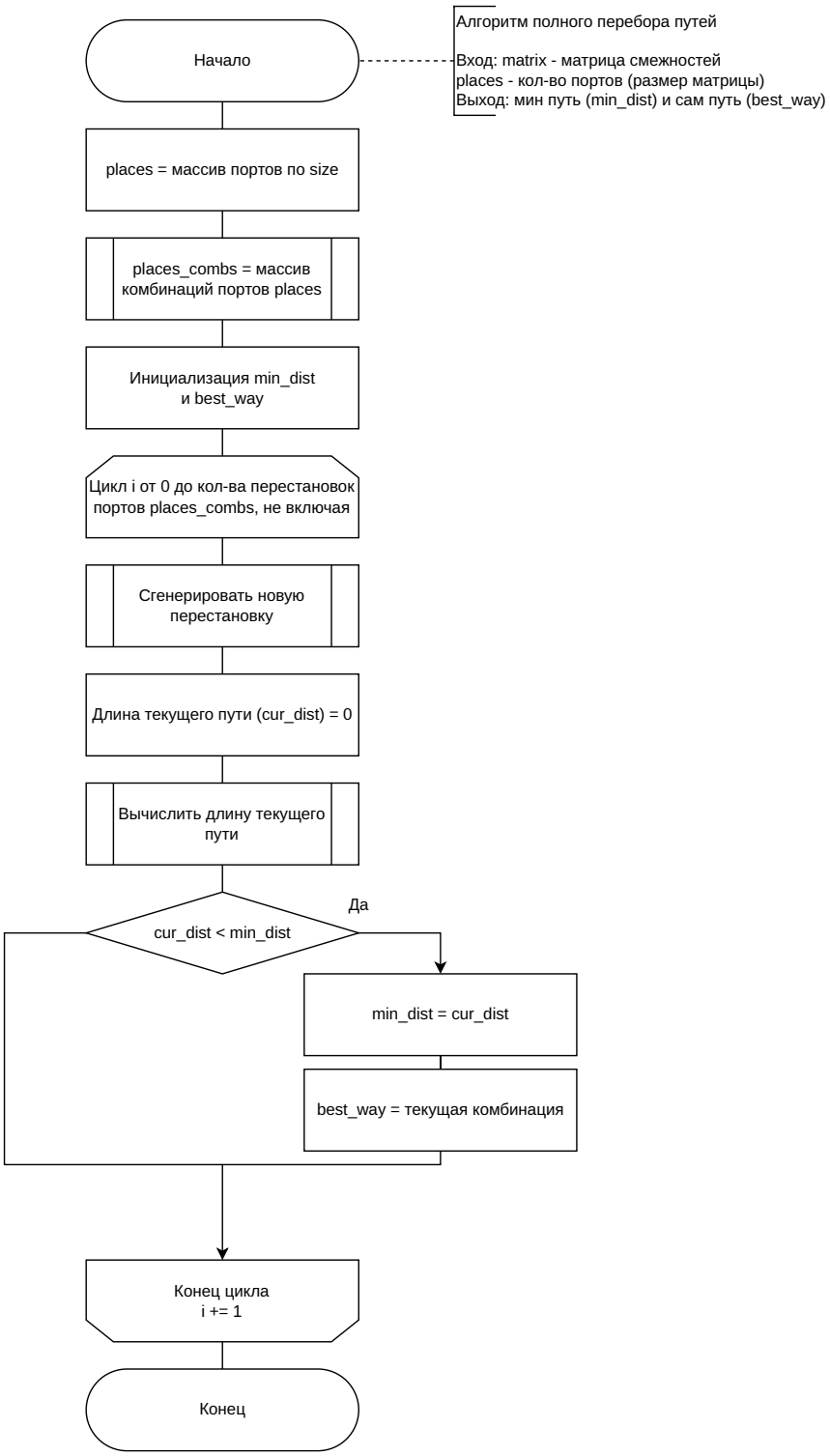

Рисунок 2.1 – Схема алгоритма полного перебора путей

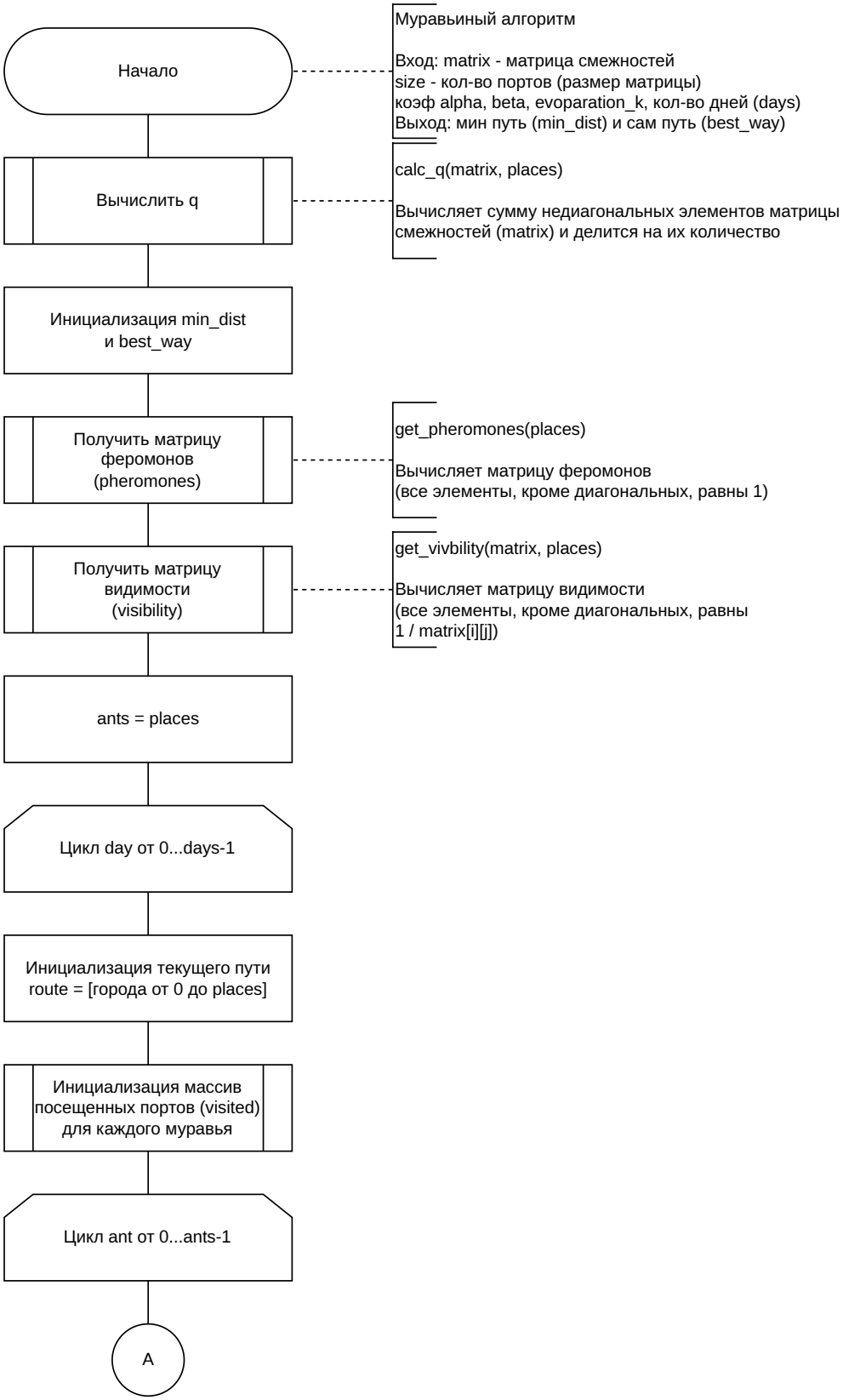

Рисунок 2.2 – Схема муравьиного алгоритма (часть 1)

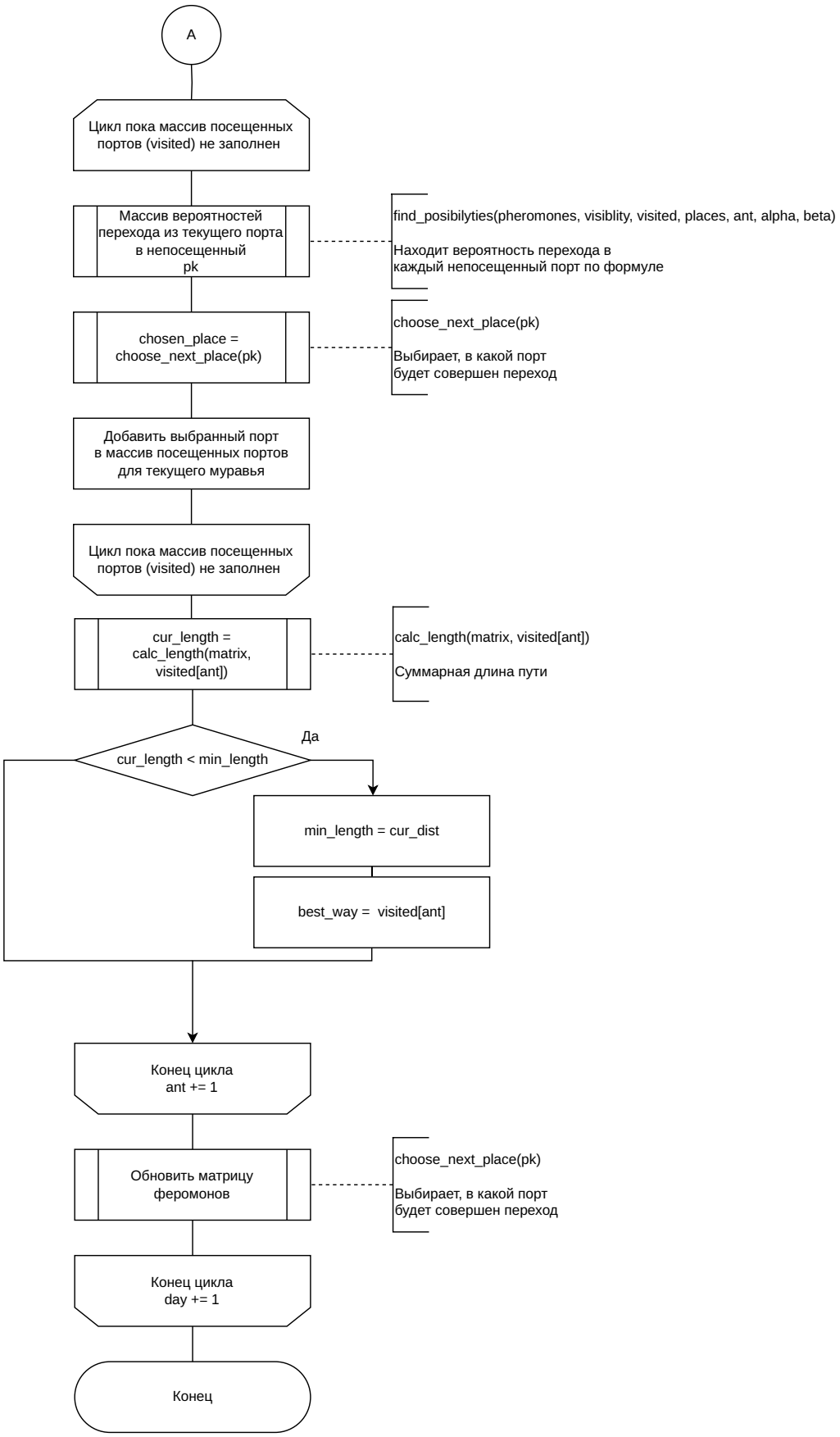

Рисунок 2.3 – Схема муравьиного алгоритма (часть 2)

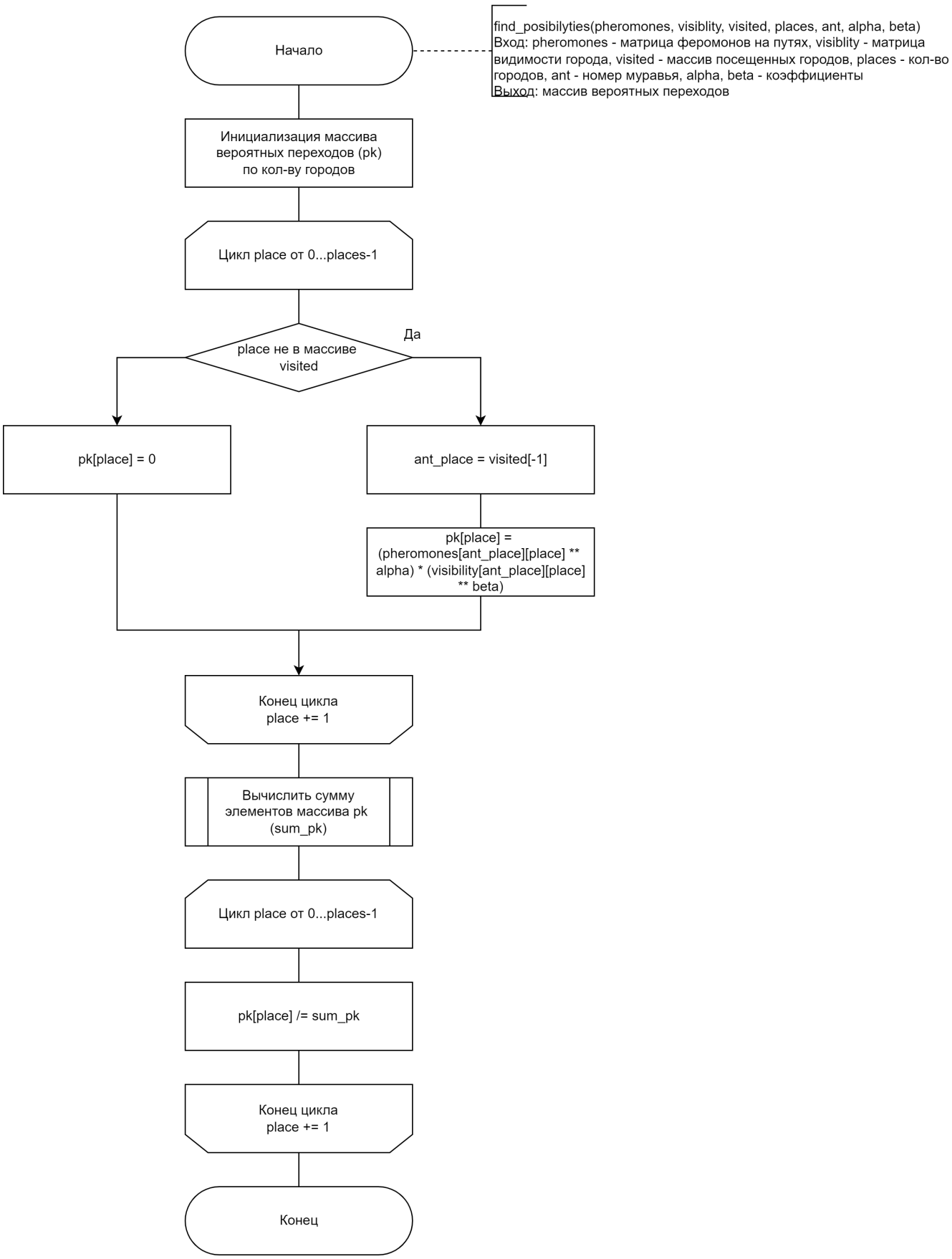

Рисунок 2.4 – Схема алгоритма нахождения массива вероятностей переходов в непосещенные порты

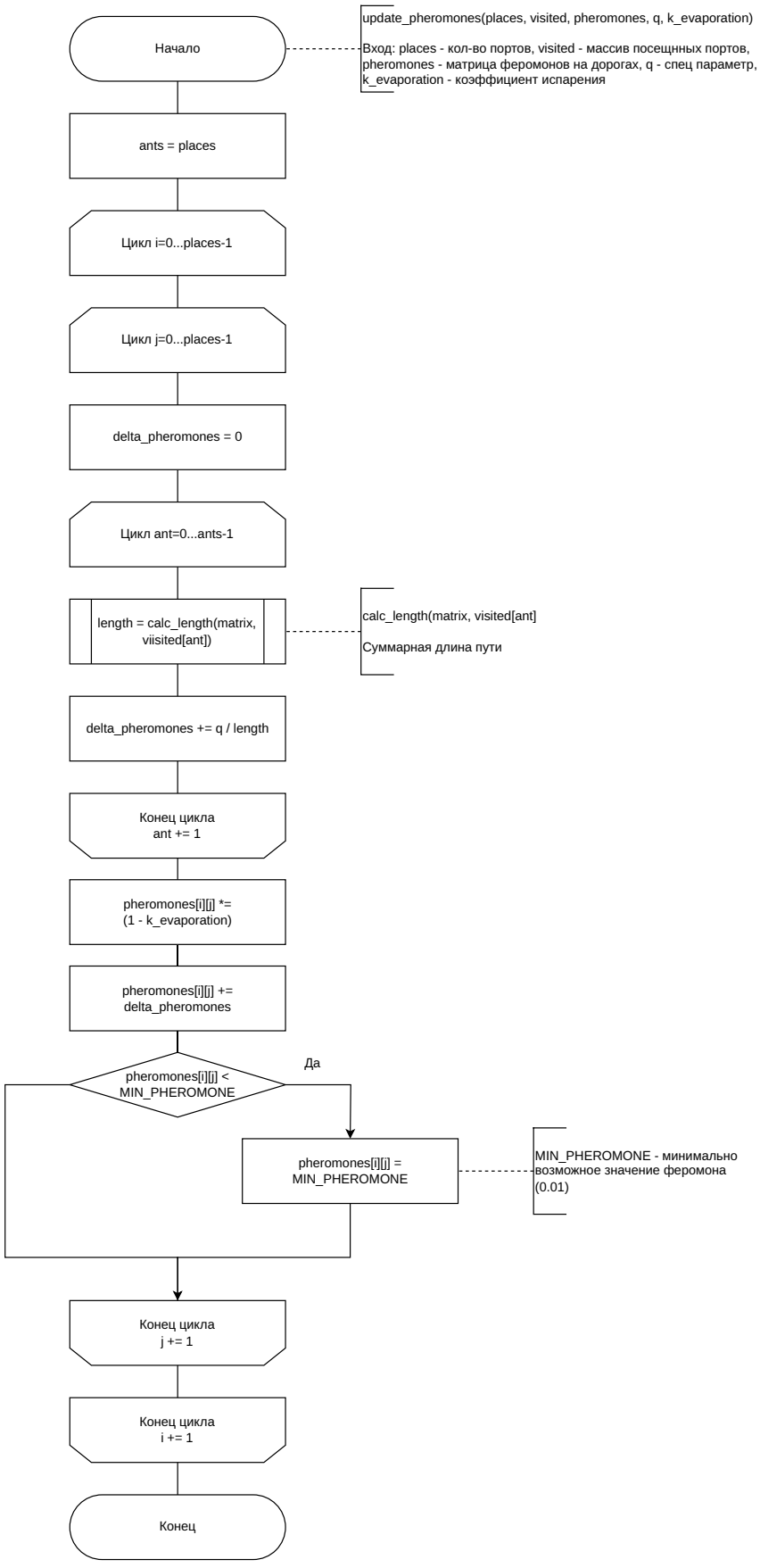

Рисунок 2.5 – Схема алгоритма обновления матрицы феромонов

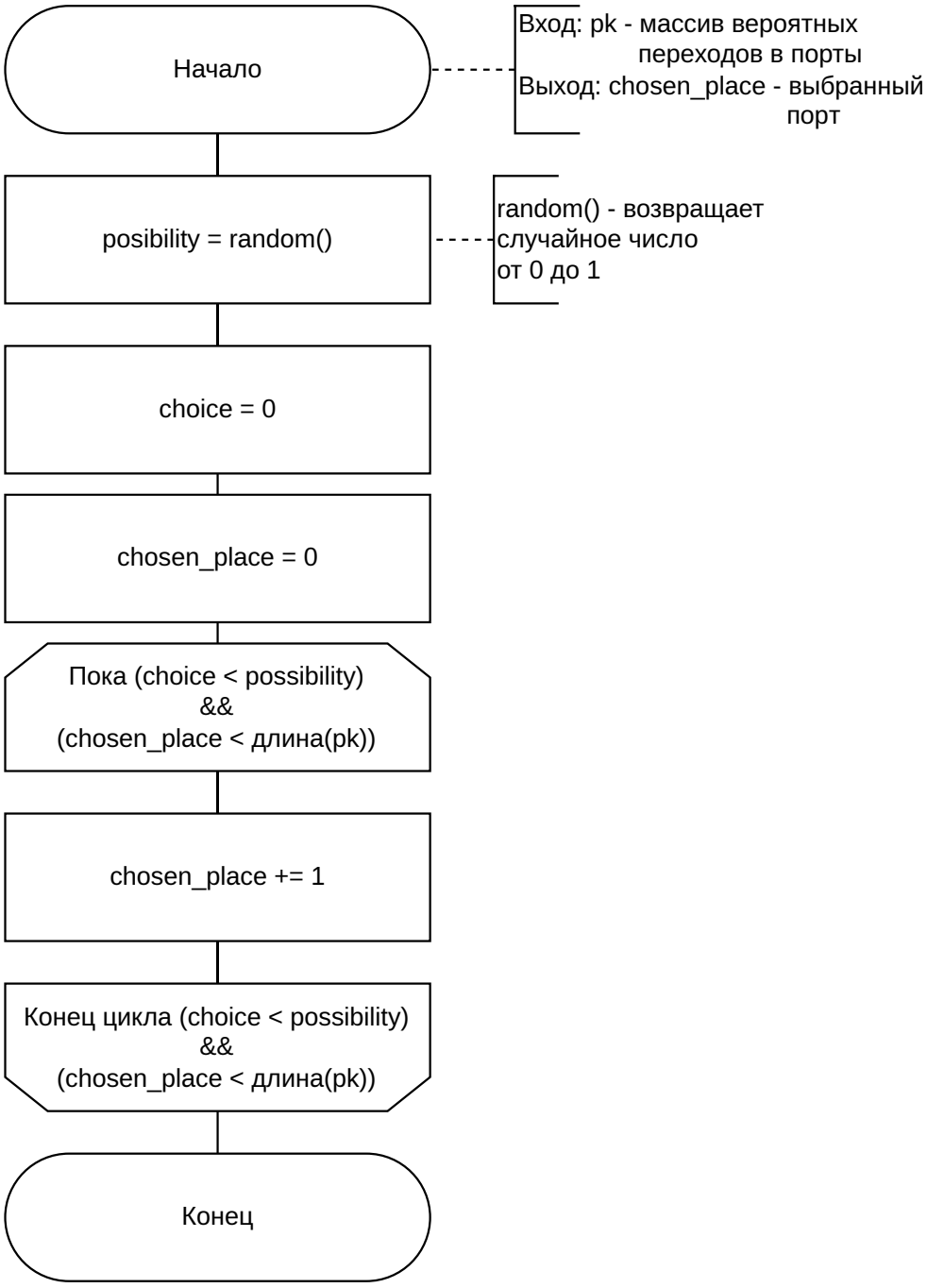

Рисунок 2.6 – Схема алгоритма выбора следующего порта

2.3 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных:

— размер матрицы смежности — целое число;

— имя файла — строка;

— матрица смежности — матрица целых чисел.

Вывод

В данном разделе была рассмотрена задача коммивояжера, а также метод полного перебора для её решения и метода на основе муравьиного алгоритма. Были представлены требования к разрабатываемому программному обеспечению.

3 Технологическая часть

В данном разделе рассмотрены средства реализации, а также представлены листинги реализаций рассматриваемых алгоритмов.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python* [3]. В текущей лабораторной работе требуется замерить процессорное время работы выполняемой программы. Инструменты для этого присутствуют в выбранном языке программирования.

Время работы было замерено с помощью функции *process_time(...)* из библиотеки *time* [4].

3.2 Сведения о файлах программы

Данная программа разбита на следующие файлы:

— *main.py* — файл, содержащий точку входа;

— *menu.py* — файл, содержащий код меню программы;

— *utils.py* — файл, содержащий служебные алгоритмы;

— *constants.py* — файл, содержащий константы программы;

— *algorythms.py* — файл, содержащий код всех алгоритмов.

3.3 Реализация алгоритмов

В листинге 3.1 представлен реализация алгоритм полного перебора путей, а в листингах 3.2–3.6 — муравьиный алгоритм и дополнительные к нему функции.

Листинг 3.1 – Реализаиция алгоритма полного перебора

```
1  def fullCombinationAlg(matrix, size):
2      places = np.arange(size)
3      placesCombinations = list()
4
5      for combination in it.permutations(places):
6          combArr = list(combination)
7          placesCombinations.append(combArr)
8
9      minDist = float("inf")
10
11     for i in range(len(placesCombinations)):
12         curDist = 0
13         for j in range(size - 1):
14             startCity = placesCombinations[i][j]
15             endCity = placesCombinations[i][j + 1]
16             curDist += matrix[startCity][endCity]
17
18         if (curDist < minDist):
19             minDist = curDist
20             bestWay = placesCombinations[i]
21
22     return minDist, bestWay
```

Листинг 3.2 – Реализаиция муравьиного алгоритма

```
def antAlgorithm(matrix, places, alpha, beta, k_evaporation,
    days, elite_ant_total):
    q = calcQ(matrix, places)
    bestWay = []
    minDist = float("inf")
    pheromones = calcPheromones(places)
    visibility = calcVisibility(matrix, places)
    ants = places
    for day in range(days):
        route = np.arange(places)
        visited = calcVisitedPlaces(route, ants)
        for ant in range(ants):
            while (len(visited[ant]) != ants):
                pk = findWays(pheromones, visibility, visited,
                    places, ant, alpha, beta)
                chosenPlace = chooseNextPlaceByPosibility(pk)
                visited[ant].append(chosenPlace - 1)

            curLength = calcLength(matrix, visited[ant])

            if (curLength < minDist):
                minDist = curLength
                bestWay = visited[ant]

        for i in range(elite_ant_total):
            visited.append(bestWay)

        pheromones = updatePheromones(matrix, places, visited,
            pheromones, q, k_evaporation)

    return minDist, bestWay
```

Листинг 3.3 – Реализация алгоритма нахождения массива вероятностей переходов в непосещенные порты

```
def findWays(pheromones, visibility, visited, places, ant,
    alpha, beta):
    pk = [0] * places

    for place in range(places):
        if place not in visited[ant]:
            ant_place = visited[ant][-1]
            pk[place] = pow(pheromones[ant_place][place],
                alpha) * \
                pow(visibility[ant_place][place], beta)
        else:
            pk[place] = 0

    sum_pk = sum(pk)

    for place in range(places):
        pk[place] /= sum_pk

    return pk
```

Листинг 3.4 – Реализация алгоритма нахождения массива вероятностей переходов в непосещенные порты

```
def calcPheromones(size):
    min_phero = 1
    pheromones = [[min_phero for i in range(size)] for j in
        range(size)]
    return pheromones
```

Листинг 3.5 – Реализация алгоритма выбора следующего порта

```
def chooseNextPlaceByPosibility(pk):
    posibility = random()
    choice = 0
    chosenPlace = 0
    while ((choice < posibility) and (chosenPlace < len(pk))):
        choice += pk[chosenPlace]
        chosenPlace += 1

    return chosenPlace
```

Листинг 3.6 – Реализация алгоритма обновления матрицы феромонов

```
def updatePheromones(matrix, places, visited, pheromones, q,
    k_evaporation):
    ants = places

    for i in range(places):
        for j in range(places):
            delta = 0
            for ant in range(ants):
                if pathInWay(i, j, visited[ant]):
                    length = calcLength(matrix, visited[ant])
                    delta += q / length

            pheromones[i][j] *= (1 - k_evaporation)
            pheromones[i][j] += delta
            if (pheromones[i][j] < MIN_PHEROMONE):
                pheromones[i][j] = MIN_PHEROMONE

    return pheromones
```

3.4 Функциональные тесты

В таблицах 3.1 – 3.2 приведены тесты для реализаций алгоритма полного перебора и муравьиного алгоритма соответственно.

Поскольку метод на основе муравьиного алгоритма не всегда находит кратчаший путь, тест будет считаться пройденным, если полученное значение отличается от эталонного не более, чем на 5. Все функциональные тесты пройдены *успешно*.

Таблица 3.1 – Функциональные тесты для реализации алгоритма полного перебора

Матрица смежности	Ожидаемый результат	Результат программы
$\begin{pmatrix} 0 & 4 & 2 & 1 & 7 \\ 1 & 0 & 3 & 7 & 2 \\ 5 & 3 & 0 & 10 & 3 \\ 8 & 6 & 7 & 0 & 9 \\ 1 & 5 & 10 & 4 & 0 \end{pmatrix}$	11, [2, 1, 4, 0, 3]	11, [2, 1, 4, 0, 3]
$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$	3, [0, 1, 2]	3, [0, 1, 2]
$\begin{pmatrix} 0 & 15 & 19 & 20 \\ 15 & 0 & 12 & 13 \\ 19 & 12 & 0 & 17 \\ 20 & 13 & 17 & 0 \end{pmatrix}$	44, [0, 1, 2, 3]	44, [0, 1, 2, 3]

Таблица 3.2 – Функциональные тесты для реализации муравьиного алгоритма

Матрица смежности	Ожидаемый результат	Результат программы
$\begin{pmatrix} 0 & 4 & 2 & 1 & 7 \\ 1 & 0 & 3 & 7 & 2 \\ 5 & 3 & 0 & 10 & 3 \\ 8 & 6 & 7 & 0 & 9 \\ 1 & 5 & 10 & 4 & 0 \end{pmatrix}$	11, [2, 1, 4, 0, 3]	11, [2, 1, 4, 0, 3]
$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$	3, [0, 1, 2]	3, [0, 1, 2]
$\begin{pmatrix} 0 & 15 & 19 & 20 \\ 15 & 0 & 12 & 13 \\ 19 & 12 & 0 & 17 \\ 20 & 13 & 17 & 0 \end{pmatrix}$	44, [0, 1, 2, 3]	44, [0, 1, 2, 3]

Вывод

Были представлены листинги всех реализаций алгоритмов — полного перебора и муравьиного. Также в данном разделе была приведена информации о выбранных средствах для разработки алгоритмов и сведения о файлах программы, проведено функциональное тестирование.

4 Исследовательская часть

В данном разделе будет приведен пример работы программы, а также проведен сравнительный анализ алгоритмов при различных ситуациях на основе полученных данных.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры по времени представлены далее.

— Процессор: Intel(R) Core(TM) i7-1165G7 CPU 2.80 ГГц.

— Количество ядер: 4 физических и 8 логических.

— Оперативная память: 15 ГБайт.

— Операционная система: Ubuntu 64-разрядная система версии 22.04.3.

При замерах времени ноутбук был включен в сеть электропитания и был нагружен только системными приложениями.

4.2 Демонстрация работы программы

На рисунке 4.1 представлен пример работы программы для обоих алгоритмов — полного перебора и муравьиного. Осуществляется выбор файла с данными, ввод коэффициентов для муравьиного алгоритма, а также выполнение алгоритма решения задачи коммивояжера методом полного перебора и методом на основе муравьиного алгоритма.

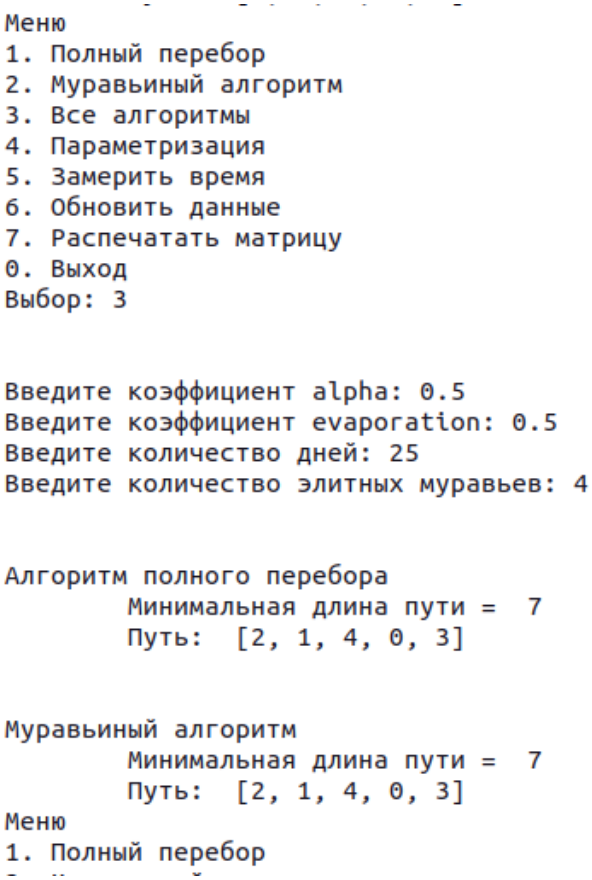

Рисунок 4.1 – Пример работы программы

4.3 Временные характеристики

Для замеров времени используется функция *process_ time(...)* из библиотеки *time* для *Python*. Функция возвращает процессорное время в секундах.

Замеры проводились для разного размера матриц, чтобы определить, когда наиболее эффективно использовать муравьиный алгоритм.

Результаты замеров приведены в таблице 4.1 (время в секундах).

Таблица 4.1 – Результаты замеров времени (в с.)

Размер матрицы	**Время, мкс**	
	Полный перебор	**Полный перебор**
2	0.000014	0.006581
3	0.000019	0.008429
4	0.000058	0.020398
5	0.000170	0.041318
6	0.001066	0.075045
7	0.007585	0.121320
8	0.075833	0.189757
9	0.785701	0.297062
10	9.015445	0.448497

На рисунке 4.2 приведен график результатов замеров времени работы реализаций алгоритмов для различных линейных размеров матриц.

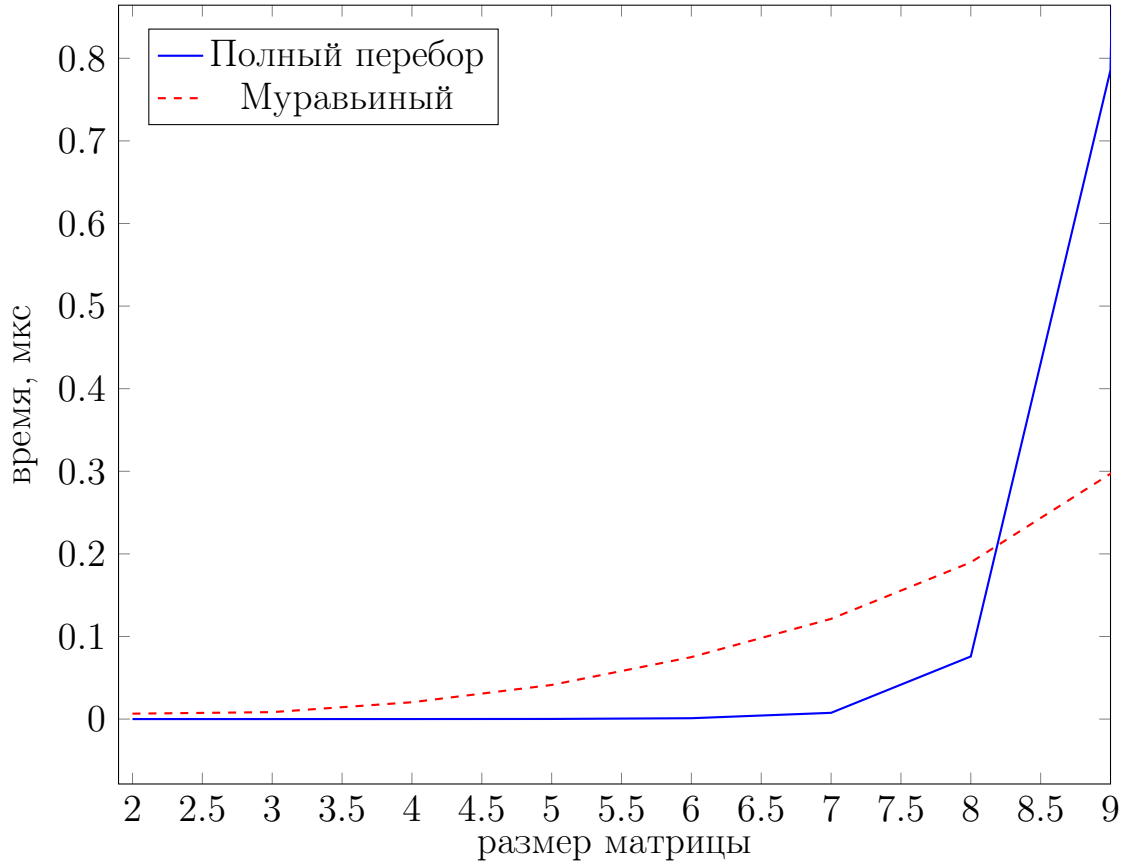

Рисунок 4.2 – Результаты замеров времени работы реализации для различных размеров матриц

4.4 Постановка эксперимента

Автоматическая параметризация была проведена на трех классах данных. Алгоритм будет запущен для набора значений $\alpha, \rho \in (0, 1)$, Elites $\in (0, 1, 3, 6), t_{max1} = 500, t_{max2} = 5000$, где t_{max1} использовался для первого класса данных, t_{max2} использовался для второго. Количество дней всегда равно 100.

Итоговая таблица значений параметризации будет состоять из следующих колонок:

— α — коэффициент жадности;

— ρ — коэффициент испарения;

— *Elites* — количество элитных муравьев;

— *Mistake* — разность полученного основанным на муравьином алгоритме методом значения и эталонного значения на данных значениях параметров, показатель качества решения.

Цель эксперимента — определить комбинацию параметров, которые позволяют решить задачу с наименьшим значением ошибки для выбранного класса данных. Качество решения зависит от погрешности измерений.

4.4.1 Класс данных 1

Класс данных 1 представляет собой 3 матрицы смежности для 10 вершин (небольшой разброс длины пути — от 10 до 50), каждая из которых представлена далее.

$$K_{1.1} = \begin{pmatrix} 0 & 22 & 11 & 30 & 25 & 16 & 41 & 28 & 26 & 35 \\ 40 & 0 & 16 & 15 & 26 & 30 & 13 & 11 & 33 & 17 \\ 24 & 23 & 0 & 29 & 25 & 18 & 12 & 42 & 31 & 35 \\ 10 & 50 & 27 & 0 & 45 & 29 & 41 & 40 & 30 & 20 \\ 31 & 29 & 14 & 14 & 0 & 29 & 26 & 21 & 37 & 21 \\ 25 & 11 & 13 & 23 & 42 & 0 & 14 & 22 & 34 & 23 \\ 12 & 28 & 39 & 37 & 45 & 41 & 0 & 23 & 32 & 20 \\ 17 & 35 & 42 & 19 & 25 & 32 & 44 & 0 & 28 & 16 \\ 38 & 47 & 18 & 30 & 24 & 16 & 32 & 33 & 0 & 22 \\ 33 & 48 & 42 & 23 & 41 & 35 & 43 & 24 & 31 & 0 \end{pmatrix} \tag{4.1}$$

$$K_{1.2} = \begin{pmatrix} 0 & 10 & 31 & 13 & 36 & 26 & 40 & 25 & 48 & 24 \\ 42 & 0 & 50 & 46 & 32 & 48 & 49 & 15 & 19 & 18 \\ 37 & 48 & 0 & 47 & 41 & 22 & 22 & 50 & 32 & 11 \\ 22 & 23 & 35 & 0 & 17 & 48 & 37 & 34 & 50 & 25 \\ 14 & 15 & 45 & 43 & 0 & 19 & 13 & 16 & 25 & 39 \\ 25 & 36 & 16 & 25 & 38 & 0 & 46 & 45 & 32 & 37 \\ 20 & 22 & 17 & 33 & 50 & 49 & 0 & 49 & 28 & 39 \\ 27 & 34 & 12 & 14 & 37 & 32 & 35 & 0 & 34 & 13 \\ 50 & 35 & 33 & 46 & 34 & 24 & 41 & 13 & 0 & 35 \\ 18 & 47 & 39 & 43 & 36 & 38 & 23 & 50 & 44 & 0 \end{pmatrix} \tag{4.2}$$

$$K_{1.3} = \begin{pmatrix} 0 & 42 & 36 & 15 & 22 & 19 & 38 & 32 & 19 & 23 \\ 38 & 0 & 26 & 27 & 36 & 19 & 36 & 16 & 26 & 18 \\ 25 & 29 & 0 & 45 & 22 & 18 & 12 & 40 & 43 & 20 \\ 11 & 22 & 21 & 0 & 35 & 49 & 16 & 34 & 24 & 19 \\ 26 & 45 & 28 & 30 & 0 & 47 & 29 & 33 & 36 & 31 \\ 37 & 50 & 17 & 18 & 43 & 0 & 39 & 45 & 34 & 18 \\ 17 & 39 & 39 & 39 & 38 & 43 & 0 & 30 & 19 & 40 \\ 48 & 35 & 29 & 24 & 39 & 31 & 42 & 0 & 40 & 27 \\ 44 & 40 & 48 & 35 & 23 & 25 & 44 & 49 & 0 & 18 \\ 25 & 40 & 11 & 41 & 16 & 28 & 20 & 46 & 17 & 0 \end{pmatrix} \tag{4.3}$$

Для данного класса данных параметризация приведена в приложении А. Для каждого набора параметров в качестве разности указано макси-

мальное полученное значение отклонения для всех матриц данного класса при указанных параметрах.

Лучшими наборами этого класса являются ($\alpha = 0.5, \rho = 0.3, \text{Elites} =$ любое), ($\alpha = 0.4, \rho = 0.7, \text{Elites} =$ любое), ($\alpha = 0.7, \rho = 0.7, \text{Elites} =$ любое), ($\alpha = 0.8, \rho = 0.2, \text{Elites} =$ любое). При этих значениях α и ρ ошибка принимает значения 0 или 1 при любом количестве элитных муравьев, из-за чего рекомендуется использовать именно эти значения.

Корреляцию между количеством элитных муравьев и погрешностью найти не удалось.

4.4.2 Класс данных 2

Класс данных 2 представляет собой 3 матрицы смежности для 10 вершин (большой разброс значений длины пути - от 100 до 999), каждая из которых представлена далее.

$$K_{2.1} = \begin{pmatrix} 0 & 551 & 637 & 618 & 475 & 983 & 397 & 818 & 507 & 173 \\ 747 & 0 & 340 & 686 & 173 & 433 & 871 & 636 & 194 & 357 \\ 599 & 405 & 0 & 108 & 737 & 555 & 145 & 527 & 428 & 329 \\ 470 & 344 & 193 & 0 & 192 & 403 & 223 & 900 & 285 & 974 \\ 632 & 613 & 841 & 994 & 0 & 547 & 644 & 375 & 590 & 437 \\ 596 & 358 & 636 & 467 & 848 & 0 & 922 & 998 & 218 & 875 \\ 742 & 962 & 424 & 858 & 219 & 481 & 0 & 596 & 320 & 117 \\ 685 & 712 & 897 & 535 & 868 & 547 & 953 & 0 & 271 & 935 \\ 388 & 110 & 571 & 563 & 454 & 537 & 258 & 406 & 0 & 399 \\ 964 & 484 & 426 & 817 & 216 & 752 & 304 & 730 & 693 & 0 \end{pmatrix} \tag{4.4}$$

$$K_{2.2} = \begin{pmatrix} 0 & 854 & 112 & 569 & 144 & 102 & 788 & 751 & 295 & 138 \\ 819 & 0 & 741 & 419 & 346 & 945 & 285 & 791 & 185 & 934 \\ 299 & 150 & 0 & 542 & 631 & 650 & 325 & 406 & 801 & 838 \\ 817 & 505 & 575 & 0 & 123 & 582 & 290 & 518 & 627 & 755 \\ 886 & 151 & 497 & 271 & 0 & 409 & 799 & 180 & 891 & 468 \\ 838 & 503 & 954 & 322 & 311 & 0 & 878 & 158 & 832 & 995 \\ 141 & 972 & 339 & 882 & 305 & 100 & 0 & 157 & 781 & 153 \\ 756 & 188 & 442 & 902 & 393 & 242 & 974 & 0 & 859 & 196 \\ 552 & 168 & 668 & 298 & 322 & 663 & 405 & 186 & 0 & 658 \\ 396 & 847 & 630 & 895 & 742 & 560 & 713 & 894 & 180 & 0 \end{pmatrix} \tag{4.5}$$

$$K_{2.3} = \begin{pmatrix} 0 & 814 & 726 & 929 & 155 & 509 & 211 & 790 & 808 & 917 \\ 317 & 0 & 739 & 648 & 955 & 766 & 390 & 560 & 900 & 734 \\ 435 & 467 & 0 & 557 & 406 & 424 & 994 & 803 & 424 & 268 \\ 324 & 896 & 988 & 0 & 712 & 877 & 455 & 435 & 866 & 478 \\ 481 & 572 & 388 & 956 & 0 & 393 & 766 & 566 & 128 & 961 \\ 690 & 710 & 278 & 591 & 587 & 0 & 293 & 461 & 124 & 280 \\ 778 & 637 & 384 & 567 & 332 & 241 & 0 & 594 & 755 & 209 \\ 988 & 611 & 661 & 598 & 400 & 409 & 956 & 0 & 517 & 229 \\ 610 & 375 & 922 & 256 & 169 & 448 & 477 & 834 & 0 & 665 \\ 235 & 829 & 888 & 327 & 888 & 965 & 566 & 722 & 599 & 0 \end{pmatrix} \tag{4.6}$$

Для данного класса данных параметризация приведена в приложении Б. Для каждого набора параметров в качестве разности указано максимальное полученное значение отклонения для всех матриц данного класса при указанных параметрах.

Лучшими наборами этого класса являются ($\alpha = 0.6, \rho = 0.2, \text{Elites} =$ любое) При этих значениях α и ρ ошибка принимает значение 0 при любом количестве элитных муравьев, из-за чего рекомендуется использовать именно эти значения.

Корреляцию между количеством элитных муравьев и погрешностью найти не удалось.

Вывод

В результате эксперимента было получено, что использование муравьиного алгоритма наиболее эффективно для матриц, размер которых превышает 8. Так, при размере матрицы равном 2, муравьиный алгоритм медленее алгоритма полного перебора в 143 раза, а при размере матрицы, равном 8, муравьиный алгоритм быстрее алгоритма полного перебора в 11 раз, а при размере в 10 – уже в 15 раз. Следовательно, при размерах матриц больше 8 следует использовать муравьиный алгоритм, но стоит учитывать, что он не гарантирует получение минимального результата при решении задачи.

Также при проведении эксперимента с классами данных было получено, что на первом классе данных муравьиный алгоритм лучше всего показывает себя при параметрах:

— $\alpha = 0.5, \rho = 0.3, \text{Elites} = \text{любое}$;

— $\alpha = 0.4, \rho = 0.7, \text{Elites} = \text{любое}$;

— $\alpha = 0.7, \rho = 0.7, \text{Elites} = \text{любое}$;

— $\alpha = 0.8, \rho = 0.2, \text{Elites} = \text{любое}$;

Для класса данных 1 рекомендуется использовать данные параметры.

Для класса данных 2 было получено, что с наименьшим значением ошибки алгоритм работает при значениии параметров $\alpha = 0.6, \rho = 0.2, \text{Elites} = \text{любое}$. Для класса данных 2 рекомендуется использовать данные параметры.

Корреляцию между количеством элитных муравьев и погрешностью найти не удалось ни для одного из классов данных.

Заключение

Поставленная цель достигнута: была выполнена параметризация метода решения задачи коммивояжера на основе муравьиного алгоритма.

В ходе выполнения лабораторной работы были решены все задачи:

1) описана задача коммивояжера;
2) описаны методы решения задачи коммивояжера — метод полного перебора и метод на основе муравьиного алгоритма;
3) приведены схемы муравьиного алгоритма и алгоритма, позволяющего решить задачу коммивояжера методом полного перебора;
4) разработан и реализован программный продукт, позволяющий решить задачу коммивояжера исследуемыми методами;
5) сравнены по времени метод полного перебора и метод на основе муравьиного алгоритма;
6) описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе.

Исходя из полученных результатов, использование муравьиного алгоритма наиболее эффективно по времени при больших размерах матриц. Так при размере матрицы, равном 2, муравьиный алгоритм меленее алгоритма полного перебора в 143 раза, а при размере матрицы, равном 9, муравьиный алгоритм быстрее алгоритма полного перебора в 11 раз, а при размере в 10 – уже в 15 раз. Следовательно, при размерах матриц больше 8 следует использовать муравьиный алгоритм, но стоит учитывать, что он не гарантирует оптимального решения, в отличие от метода полного перебора.

Список использованных источников

1 В.О. Борознов. Исследование решения задачи коммивояжера. — АГТУ, Вестник Астраханского государственного технического университета. [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/issledovanie-resheniya-zadachi-kommivoyazhera/viewer (дата обращения: 23.01.2023).

2 С.С. Семёнов, А.В. Педан, В.С. Воловиков [и др.]. Анализ трудоёмкости различных алгоритмических подходов для решения задачи коммивояжёра — ООО «Корпорация «Интел Групп» [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/analiz-trudoemkosti-razlichnyh-algoritmicheskih-podhodov-dlya-\resheniya-zadachi-kommivoyazhera (дата обращения: 23.01.2023).

3 Welcome to Python [Электронный ресурс]. Режим доступа: https://www.python.org (дата обращения: 23.01.2023).

4 time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html#functions (дата обращения: 23.01.2023).

Приложение А

Таблица 4.2 – Параметризация для класса данных 1, Elites — количество элитных муравьев, Mistake — ошибочность полученного результата

α	ρ	Elites	Mistake
0.1	0.1	1	11
0.1	0.1	3	1
0.1	0.1	6	7
0.1	0.2	0	9
0.1	0.2	1	6
0.1	0.2	3	0
0.1	0.2	6	0
0.1	0.3	0	13
0.1	0.3	1	1
0.1	0.3	3	11
0.1	0.3	6	11
0.1	0.4	0	0
0.1	0.4	1	7
0.1	0.4	3	1
0.1	0.4	6	1
0.1	0.5	0	0
0.1	0.5	1	7
0.1	0.5	3	8
0.1	0.5	6	18
0.1	0.6	0	7
0.1	0.6	1	1
0.1	0.6	3	7
0.1	0.6	6	5
0.1	0.7	0	1
0.1	0.7	1	8
0.1	0.7	3	5
0.1	0.7	6	7
0.1	0.8	0	1

0.1	0.8	1	16
0.1	0.8	3	13
0.1	0.8	6	6
0.2	0.1	0	8
0.2	0.1	1	13
0.2	0.1	3	1
0.2	0.1	6	0
0.2	0.2	0	5
0.2	0.2	1	5
0.2	0.2	3	1
0.2	0.2	6	9
0.2	0.3	0	8
0.2	0.3	1	1
0.2	0.3	3	10
0.2	0.3	6	0
0.2	0.4	0	6
0.2	0.4	1	5
0.2	0.4	3	6
0.2	0.4	6	1
0.2	0.5	0	1
0.2	0.5	1	8
0.2	0.5	3	9
0.2	0.5	6	6
0.2	0.6	0	6
0.2	0.6	1	7
0.2	0.6	3	9
0.2	0.6	6	0
0.2	0.7	0	5
0.2	0.7	1	0
0.2	0.7	3	6
0.2	0.7	6	5
0.2	0.8	0	0
0.2	0.8	1	1

0.2	0.8	3	1
0.2	0.8	6	1
0.3	0.1	0	8
0.3	0.1	1	8
0.3	0.1	3	5
0.3	0.1	6	9
0.3	0.2	0	1
0.3	0.2	1	5
0.3	0.2	3	7
0.3	0.2	6	8
0.3	0.3	0	11
0.3	0.3	1	14
0.3	0.3	3	7
0.3	0.3	6	10
0.3	0.4	0	0
0.3	0.4	1	1
0.3	0.4	3	5
0.3	0.4	6	6
0.3	0.5	0	5
0.3	0.5	1	10
0.3	0.5	3	0
0.3	0.5	6	7
0.3	0.6	0	8
0.3	0.6	1	6
0.3	0.6	3	5
0.3	0.6	6	10
0.3	0.7	0	7
0.3	0.7	1	0
0.3	0.7	3	7
0.3	0.7	6	0
0.3	0.8	0	1
0.3	0.8	1	6
0.3	0.8	3	10

0.3	0.8	6	1
0.4	0.1	0	6
0.4	0.1	1	7
0.4	0.1	3	5
0.4	0.1	6	9
0.4	0.2	0	1
0.4	0.2	1	8
0.4	0.2	3	1
0.4	0.2	6	1
0.4	0.3	0	0
0.4	0.3	1	8
0.4	0.3	3	0
0.4	0.3	6	7
0.4	0.4	0	5
0.4	0.4	1	8
0.4	0.4	3	0
0.4	0.4	6	0
0.4	0.5	0	0
0.4	0.5	1	8
0.4	0.5	3	7
0.4	0.5	6	6
0.4	0.6	0	0
0.4	0.6	1	1
0.4	0.6	3	8
0.4	0.6	6	8
0.4	0.7	0	0
0.4	0.7	1	1
0.4	0.7	3	0
0.4	0.7	6	0
0.4	0.8	0	17
0.4	0.8	1	1
0.4	0.8	3	0
0.4	0.8	6	7

0.5	0.1	0	7
0.5	0.1	1	7
0.5	0.1	3	8
0.5	0.1	6	5
0.5	0.2	0	7
0.5	0.2	1	10
0.5	0.2	3	1
0.5	0.2	6	1
0.5	0.3	0	0
0.5	0.3	1	0
0.5	0.3	3	0
0.5	0.3	6	1
0.5	0.4	0	0
0.5	0.4	1	9
0.5	0.4	3	1
0.5	0.4	6	10
0.5	0.5	0	5
0.5	0.5	1	5
0.5	0.5	3	0
0.5	0.5	6	9
0.5	0.6	0	7
0.5	0.6	1	6
0.5	0.6	3	16
0.5	0.6	6	0
0.5	0.7	0	6
0.5	0.7	1	8
0.5	0.7	3	7
0.5	0.7	6	7
0.5	0.8	0	8
0.5	0.8	1	1
0.5	0.8	3	7
0.5	0.8	6	0
0.6	0.1	0	0

0.6	0.1	1	1
0.6	0.1	3	7
0.6	0.1	6	0
0.6	0.2	0	0
0.6	0.2	1	7
0.6	0.2	3	0
0.6	0.2	6	0
0.6	0.3	0	6
0.6	0.3	1	0
0.6	0.3	3	0
0.6	0.3	6	8
0.6	0.4	0	0
0.6	0.4	1	0
0.6	0.4	3	9
0.6	0.4	6	7
0.6	0.5	0	8
0.6	0.5	1	0
0.6	0.5	3	0
0.6	0.5	6	8
0.6	0.6	0	1
0.6	0.6	1	9
0.6	0.6	3	0
0.6	0.6	6	5
0.6	0.7	0	5
0.6	0.7	1	5
0.6	0.7	3	7
0.6	0.7	6	7
0.6	0.8	0	7
0.6	0.8	1	10
0.6	0.8	3	9
0.6	0.8	6	1
0.7	0.1	0	7
0.7	0.1	1	8

0.7	0.1	3	7
0.7	0.1	6	1
0.7	0.2	0	9
0.7	0.2	1	5
0.7	0.2	3	10
0.7	0.2	6	0
0.7	0.3	0	8
0.7	0.3	1	9
0.7	0.3	3	6
0.7	0.3	6	6
0.7	0.4	0	5
0.7	0.4	1	0
0.7	0.4	3	0
0.7	0.4	6	6
0.7	0.5	0	5
0.7	0.5	1	0
0.7	0.5	3	0
0.7	0.5	6	8
0.7	0.6	0	1
0.7	0.6	1	0
0.7	0.6	3	7
0.7	0.6	6	1
0.7	0.7	0	8
0.7	0.7	1	0
0.7	0.7	3	0
0.7	0.7	6	0
0.7	0.8	0	1
0.7	0.8	1	9
0.7	0.8	3	1
0.7	0.8	6	5
0.8	0.1	0	7
0.8	0.1	1	7
0.8	0.1	3	9

0.8	0.1	6	9
0.8	0.2	0	0
0.8	0.2	1	0
0.8	0.2	3	0
0.8	0.2	6	0
0.8	0.3	0	14
0.8	0.3	1	8
0.8	0.3	3	0
0.8	0.3	6	7
0.8	0.4	0	0
0.8	0.4	1	8
0.8	0.4	3	0
0.8	0.4	6	7
0.8	0.5	0	1
0.8	0.5	1	0
0.8	0.5	3	10
0.8	0.5	6	11
0.8	0.6	0	5
0.8	0.6	1	1
0.8	0.6	3	8
0.8	0.6	6	8
0.8	0.7	0	7
0.8	0.7	1	7
0.8	0.7	3	11
0.8	0.7	6	12
0.8	0.8	0	1
0.8	0.8	1	0
0.8	0.8	3	8
0.8	0.8	6	5
0.9	0.1	0	0
0.9	0.1	1	0
0.9	0.1	3	7
0.9	0.1	6	13

0.9	0.2	0	8
0.9	0.2	1	7
0.9	0.2	3	1
0.9	0.2	6	0
0.9	0.3	0	7
0.9	0.3	1	21
0.9	0.3	3	10
0.9	0.3	6	9
0.9	0.4	0	8
0.9	0.4	1	0
0.9	0.4	3	0
0.9	0.4	6	8
0.9	0.5	0	5
0.9	0.5	1	8
0.9	0.5	3	7
0.9	0.5	6	5
0.9	0.6	0	8
0.9	0.6	1	1
0.9	0.6	3	5
0.9	0.6	6	16
0.9	0.7	0	0
0.9	0.7	1	5
0.9	0.7	3	6
0.9	0.7	6	9
0.9	0.8	0	8
0.9	0.8	1	9
0.9	0.8	3	16
0.9	0.8	6	5

Приложение Б

Таблица 4.3 – Параметризация для класса данных 2, Elites — количество элитных муравьев, Mistake — ошибочность полученного результата

α	ρ	Elites	Mistake
0.1	0.1	0	29
0.1	0.1	1	117
0.1	0.1	3	0
0.1	0.1	6	165
0.1	0.2	0	70
0.1	0.2	1	106
0.1	0.2	3	204
0.1	0.2	6	70
0.1	0.3	0	70
0.1	0.3	1	13
0.1	0.3	3	70
0.1	0.3	6	70
0.1	0.4	0	106
0.1	0.4	1	0
0.1	0.4	3	124
0.1	0.4	6	125
0.1	0.5	0	70
0.1	0.5	1	70
0.1	0.5	3	172
0.1	0.5	6	13
0.1	0.6	0	0
0.1	0.6	1	106
0.1	0.6	3	29
0.1	0.6	6	0
0.1	0.7	0	13
0.1	0.7	1	106
0.1	0.7	3	0
0.1	0.7	6	13

0.1	0.8	0	0
0.1	0.8	1	70
0.1	0.8	3	0
0.1	0.8	6	29
0.2	0.1	0	70
0.2	0.1	1	29
0.2	0.1	3	0
0.2	0.1	6	13
0.2	0.2	0	105
0.2	0.2	1	106
0.2	0.2	3	0
0.2	0.2	6	70
0.2	0.3	0	70
0.2	0.3	1	70
0.2	0.3	3	106
0.2	0.3	6	70
0.2	0.4	0	130
0.2	0.4	1	130
0.2	0.4	3	0
0.2	0.4	6	13
0.2	0.5	0	0
0.2	0.5	1	70
0.2	0.5	3	0
0.2	0.5	6	124
0.2	0.6	0	124
0.2	0.6	1	165
0.2	0.6	3	105
0.2	0.6	6	106
0.2	0.7	0	0
0.2	0.7	1	189
0.2	0.7	3	70
0.2	0.7	6	117
0.2	0.8	0	70

0.2	0.8	1	0
0.2	0.8	3	117
0.2	0.8	6	0
0.3	0.1	0	125
0.3	0.1	1	125
0.3	0.1	3	105
0.3	0.1	6	106
0.3	0.2	0	163
0.3	0.2	1	172
0.3	0.2	3	105
0.3	0.2	6	172
0.3	0.3	0	13
0.3	0.3	1	174
0.3	0.3	3	0
0.3	0.3	6	70
0.3	0.4	0	172
0.3	0.4	1	106
0.3	0.4	3	70
0.3	0.4	6	172
0.3	0.5	0	13
0.3	0.5	1	13
0.3	0.5	3	0
0.3	0.5	6	105
0.3	0.6	0	117
0.3	0.6	1	70
0.3	0.6	3	130
0.3	0.6	6	145
0.3	0.7	0	260
0.3	0.7	1	70
0.3	0.7	3	174
0.3	0.7	6	172
0.3	0.8	0	106
0.3	0.8	1	125

0.3	0.8	3	29
0.3	0.8	6	13
0.4	0.1	0	130
0.4	0.1	1	0
0.4	0.1	3	106
0.4	0.1	6	117
0.4	0.2	0	70
0.4	0.2	1	106
0.4	0.2	3	0
0.4	0.2	6	124
0.4	0.3	0	70
0.4	0.3	1	125
0.4	0.3	3	106
0.4	0.3	6	179
0.4	0.4	0	124
0.4	0.4	1	13
0.4	0.4	3	0
0.4	0.4	6	117
0.4	0.5	0	70
0.4	0.5	1	70
0.4	0.5	3	106
0.4	0.5	6	13
0.4	0.6	0	13
0.4	0.6	1	117
0.4	0.6	3	13
0.4	0.6	6	0
0.4	0.7	0	70
0.4	0.7	1	165
0.4	0.7	3	13
0.4	0.7	6	29
0.4	0.8	0	130
0.4	0.8	1	106
0.4	0.8	3	124

0.4	0.8	6	13
0.5	0.1	0	105
0.5	0.1	1	105
0.5	0.1	3	29
0.5	0.1	6	117
0.5	0.2	0	29
0.5	0.2	1	117
0.5	0.2	3	13
0.5	0.2	6	0
0.5	0.3	0	105
0.5	0.3	1	70
0.5	0.3	3	105
0.5	0.3	6	13
0.5	0.4	0	124
0.5	0.4	1	13
0.5	0.4	3	213
0.5	0.4	6	13
0.5	0.5	0	29
0.5	0.5	1	106
0.5	0.5	3	106
0.5	0.5	6	145
0.5	0.6	0	188
0.5	0.6	1	0
0.5	0.6	3	70
0.5	0.6	6	13
0.5	0.7	0	130
0.5	0.7	1	29
0.5	0.7	3	0
0.5	0.7	6	106
0.5	0.8	0	70
0.5	0.8	1	13
0.5	0.8	3	0
0.5	0.8	6	117

0.6	0.1	0	163
0.6	0.1	1	0
0.6	0.1	3	130
0.6	0.1	6	70
0.6	0.2	0	13
0.6	0.2	1	0
0.6	0.2	3	0
0.6	0.2	6	0
0.6	0.3	0	0
0.6	0.3	1	29
0.6	0.3	3	174
0.6	0.3	6	13
0.6	0.4	0	13
0.6	0.4	1	70
0.6	0.4	3	0
0.6	0.4	6	70
0.6	0.5	0	13
0.6	0.5	1	70
0.6	0.5	3	0
0.6	0.5	6	13
0.6	0.6	0	0
0.6	0.6	1	106
0.6	0.6	3	0
0.6	0.6	6	179
0.6	0.7	0	0
0.6	0.7	1	189
0.6	0.7	3	70
0.6	0.7	6	70
0.6	0.8	0	29
0.6	0.8	1	13
0.6	0.8	3	0
0.6	0.8	6	0
0.7	0.1	0	29

0.7	0.1	1	130
0.7	0.1	3	13
0.7	0.1	6	117
0.7	0.2	0	70
0.7	0.2	1	106
0.7	0.2	3	29
0.7	0.2	6	172
0.7	0.3	0	125
0.7	0.3	1	0
0.7	0.3	3	0
0.7	0.3	6	13
0.7	0.4	0	174
0.7	0.4	1	105
0.7	0.4	3	0
0.7	0.4	6	106
0.7	0.5	0	0
0.7	0.5	1	170
0.7	0.5	3	125
0.7	0.5	6	13
0.7	0.6	0	125
0.7	0.6	1	0
0.7	0.6	3	13
0.7	0.6	6	0
0.7	0.7	0	165
0.7	0.7	1	170
0.7	0.7	3	70
0.7	0.7	6	70
0.7	0.8	0	172
0.7	0.8	1	170
0.7	0.8	3	13
0.7	0.8	6	29
0.8	0.1	0	0
0.8	0.1	1	0

0.8	0.1	3	165
0.8	0.1	6	165
0.8	0.2	0	29
0.8	0.2	1	70
0.8	0.2	3	125
0.8	0.2	6	172
0.8	0.3	0	124
0.8	0.3	1	117
0.8	0.3	3	125
0.8	0.3	6	70
0.8	0.4	0	172
0.8	0.4	1	106
0.8	0.4	3	0
0.8	0.4	6	170
0.8	0.5	0	0
0.8	0.5	1	105
0.8	0.5	3	29
0.8	0.5	6	0
0.8	0.6	0	29
0.8	0.6	1	201
0.8	0.6	3	106
0.8	0.6	6	13
0.8	0.7	0	0
0.8	0.7	1	70
0.8	0.7	3	70
0.8	0.7	6	234
0.8	0.8	0	70
0.8	0.8	1	13
0.8	0.8	3	29
0.8	0.8	6	0
0.9	0.1	0	70
0.9	0.1	1	117
0.9	0.1	3	188

0.9	0.1	6	0
0.9	0.2	0	0
0.9	0.2	1	188
0.9	0.2	3	392
0.9	0.2	6	201
0.9	0.3	0	200
0.9	0.3	1	124
0.9	0.3	3	213
0.9	0.3	6	130
0.9	0.4	0	0
0.9	0.4	1	188
0.9	0.4	3	179
0.9	0.4	6	106
0.9	0.5	0	172
0.9	0.5	1	0
0.9	0.5	3	145
0.9	0.5	6	29
0.9	0.6	0	360
0.9	0.6	1	125
0.9	0.6	3	70
0.9	0.6	6	215
0.9	0.7	0	174
0.9	0.7	1	0
0.9	0.7	3	117
0.9	0.7	6	283
0.9	0.8	0	70
0.9	0.8	1	229
0.9	0.8	3	261
0.9	0.8	6	204