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, искусственный интеллект и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

Отчет по лабораторной работе №6

по курсу «Анализ алгоритмов»

«Муравьиный алгоритм»

Студент группы **ИУ7-54Б** _____ **Чепиго Д.С.**

(Подпись, дата) (И.О. Фамилия)

Преподаватели _____ **Волкова Л.Л.**

(Подпись, дата) (И.О. Фамилия)

_____ **Строганов Ю.В.**

(Подпись, дата) (И.О. Фамилия)

***2022** г.*

Содержание

ВВЕДЕНИЕ

Задача коммивояжёра – одна из самых известных задач комбинаторной оптимизации, заключающаяся в поиске самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город [1]. Такая задача может быть решены при помощи полного перебора вариантов и эвристических алгоритмов. Алгоритмы, основанные на использовании эвристического метода, не всегда приводят к оптимальным решениям. Однако для их применения на практике достаточно, чтобы ошибка прогнозирования не превышала допустимого значения.

Целью данной лабораторной работы является сравнительный анализ метода полного перебора и эвристического метода на базе муравьиного алгоритма.

Задачи лабораторной работы:

— описать схемой алгоритма и реализовать метод полного перебора для решения задачи коммивояжёра.
— описать схемой алгоритма и реализовать метод решения задачи коммивояжёра на основе муравьиного алгоритма;
— выполнить оценку трудоёмкости описанных алгоритмов;
— провести сравнительный анализ двух рассмотренных методов решения задачи коммивояжёра;
— подготовить отчет по лабораторной работе.

1 Аналитическая часть

1.1 Алгоритм полного перебора

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

1.2 Муравьиный алгоритм

Муравьиный алгоритм – метод решения задач коммивояжёра на основании моделирования поведения колонии муравьев [2]. Каждый муравей определяет для себя маршрут, который необходимо пройти на основе феромона, который он получает во время прохождения, каждый муравей оставляет феромон на своем пути, чтобы остальные муравьи по нему ориентировались. В результате при прохождении каждым муравьем различного маршрута наибольшее число феромона остается на оптимальном пути.

Муравьи действуют согласно правилам:

— муравей запоминает посещенные города, причем каждый город может быть посещен только один раз. Обозначим через $J_{i,k}$ список городов, которые посетил муравей k, находящийся в городе i;

— муравей обладает видимостью η_{ij} - эвристическим желанием посетить город j, если муравей находится в городе i, причем

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

где D_{ij} — стоимость пути из города i в город j;

— муравей может улавливать след феромона - специального химического вещества. Число феромона на пути из города i в город j - τ_{ij}.

Муравей выполняет следующую последовательность действий, пока не

посетит все города:

— выбирает следующий город назначения, основываясь на вероятностно-пропорциональном правиле (2), в котором учитываются видимость и число феромона:

$$P_{ij,k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l=1}^{m}\tau_{il}^{\alpha}\eta_{il}^{\beta}}, \text{если город j необходимо посетить;} \\ 0, \text{иначе,} \end{cases} \quad (2)$$

где α - параметр влияния феромона, β - параметр влияния видимости пути, τ_{ij} - число феромона на ребре (ij), η_{ij} - эвристическое желание посетить город j, если муравей находится в городе i. Выбор города является вероятностным, данное правило определяет ширину зоны города j, в общую зону всех городов $J_{i,k}$ бросается случайное число, которое и определяет выбор муравья;

— муравей проходит путь (ij) и оставляет на нем феромон.

Информация о числе феромона на пути используется другими муравьями для выбора пути. Те муравьи, которые случайно выберут кратчайший путь, будут быстрее его проходить, и за несколько передвижений он будет более обогащен феромоном. Следующие муравьи будут предпочитать именно этот путь, продолжая обогащать его феромоном.

После прохождения маршрутов всеми муравьями значение феромона на путях обновляется в соответствии со следующим правилом (3):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}, \quad (3)$$

где ρ - коэффициент испарения. Чтобы найденное локальное решение не было единственным, моделируется испарение феромона.

При этом

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \tau_{ij,k}, \quad (4)$$

где m - число муравьев,

$$\Delta\tau_{ij,k} = \begin{cases} Q/L_k, \text{если k-ый муравей прошел путь (i,j);} \\ 0, \text{иначе.} \end{cases} \quad (5)$$

2 Конструкторская часть

Разработка алгоритмов

На рисунке 1 приведена схема алгоритма решения задачи коммивояжера полным перебором. Схемы муравьиного алгоритма приведена на рисунке 2, вспомогательные функции данного алгоритма показаны на рисунках 4-6.

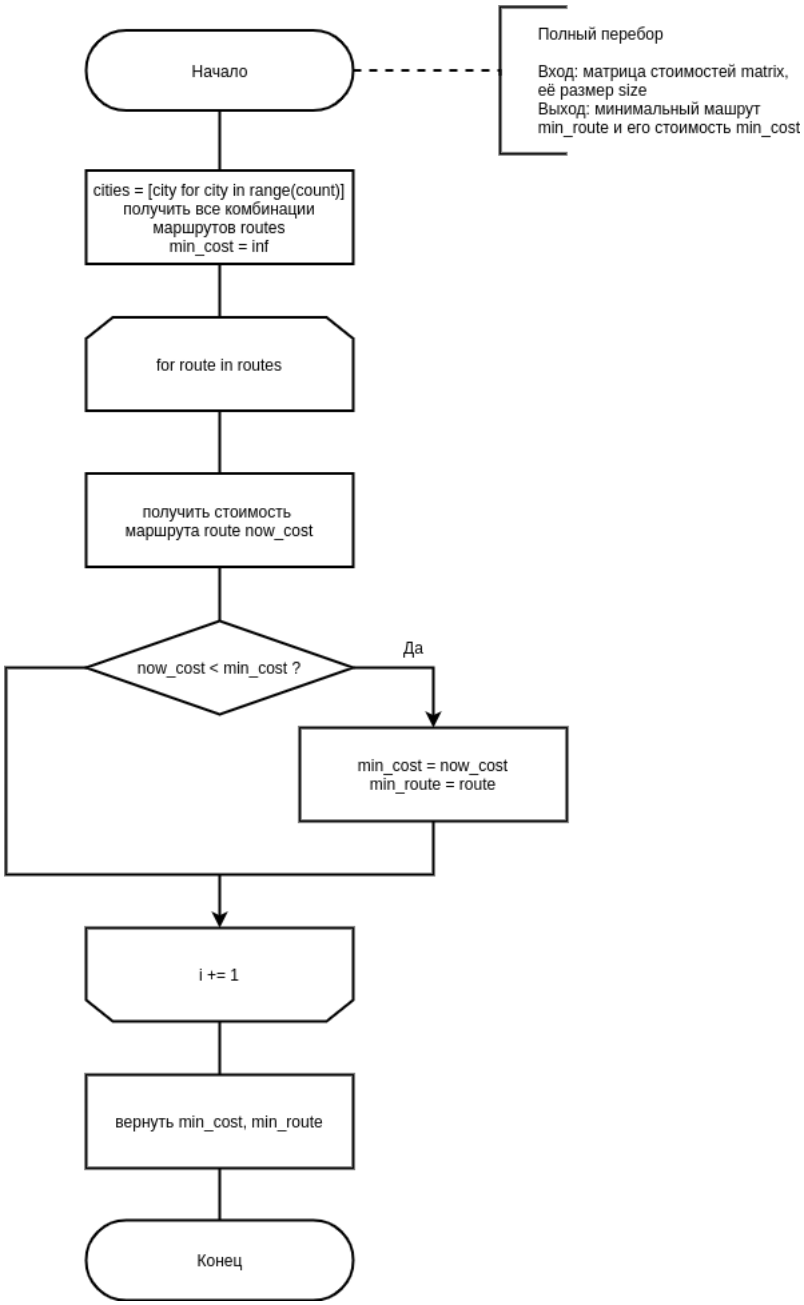

Рисунок 1 – Полный перебор

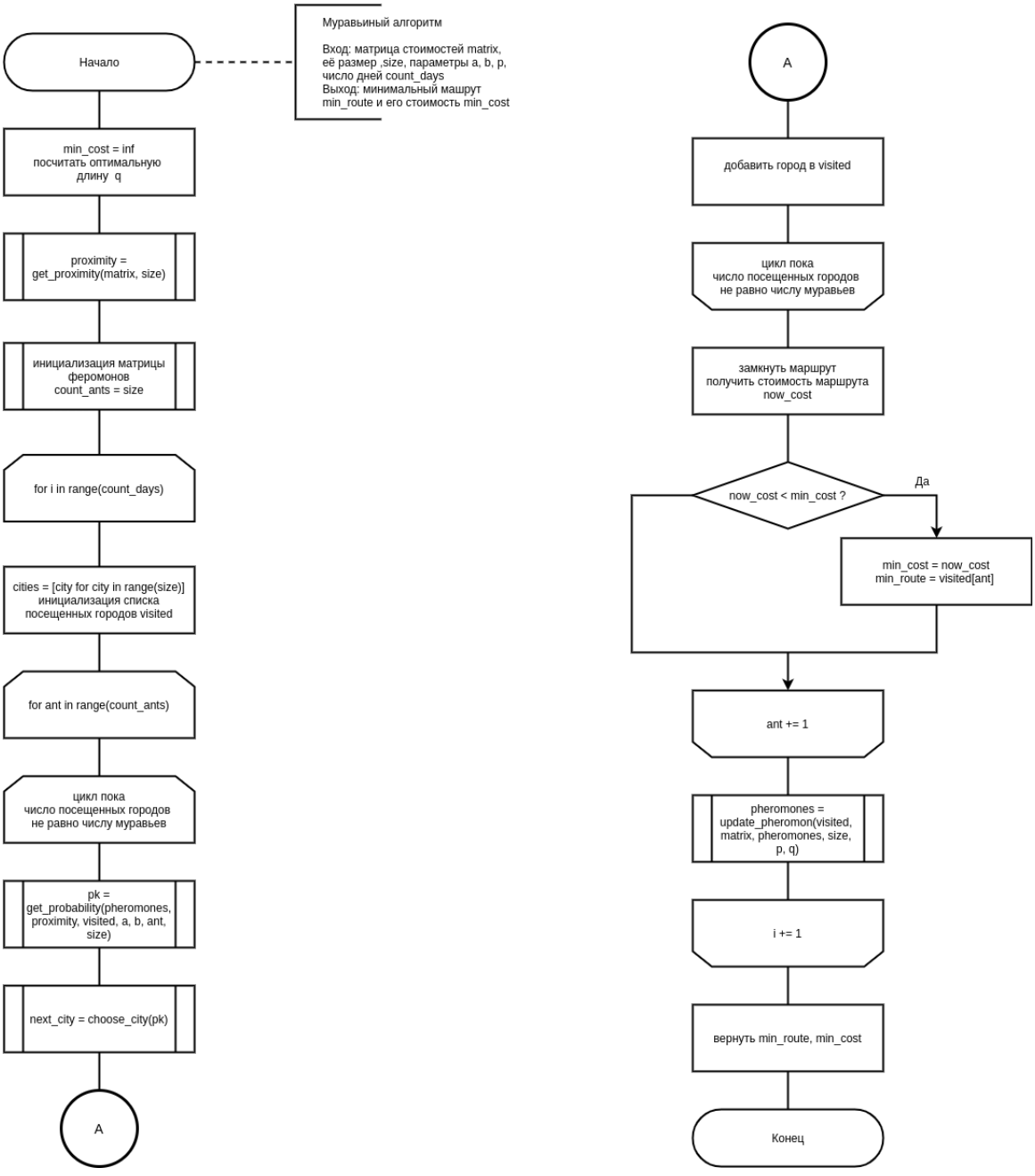

Рисунок 2 – Муравьиный алгоритм

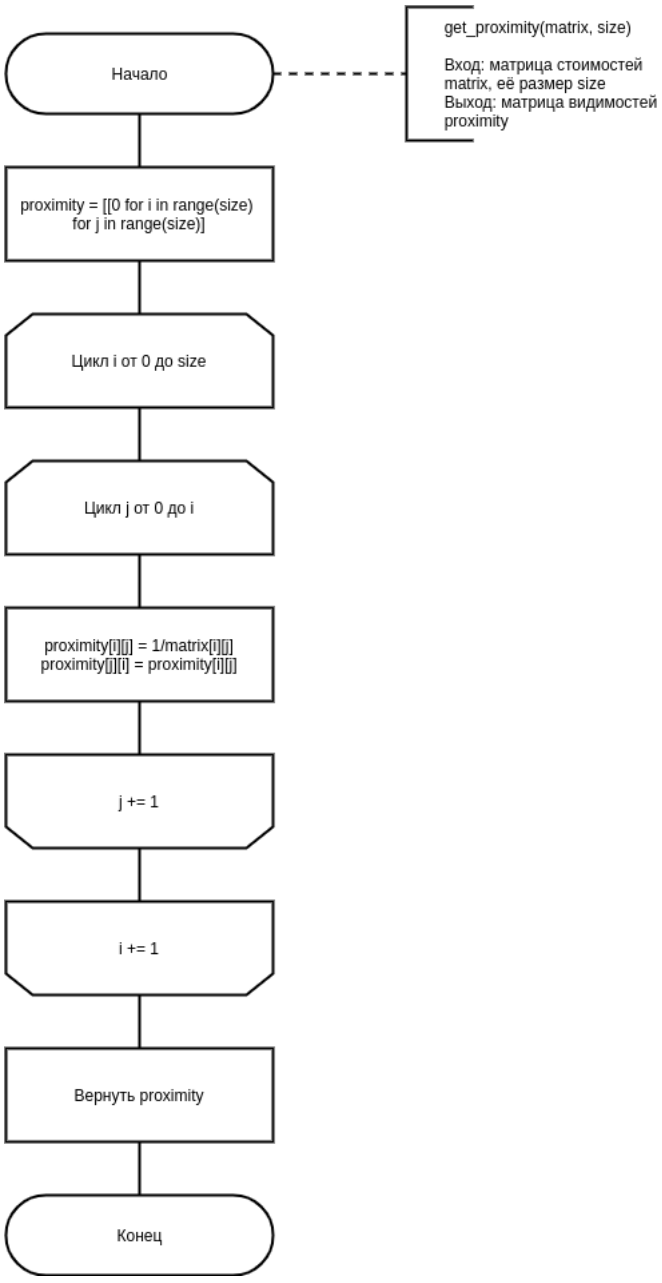

Рисунок 3 – Алгоритм вычисления матрицы видимостей

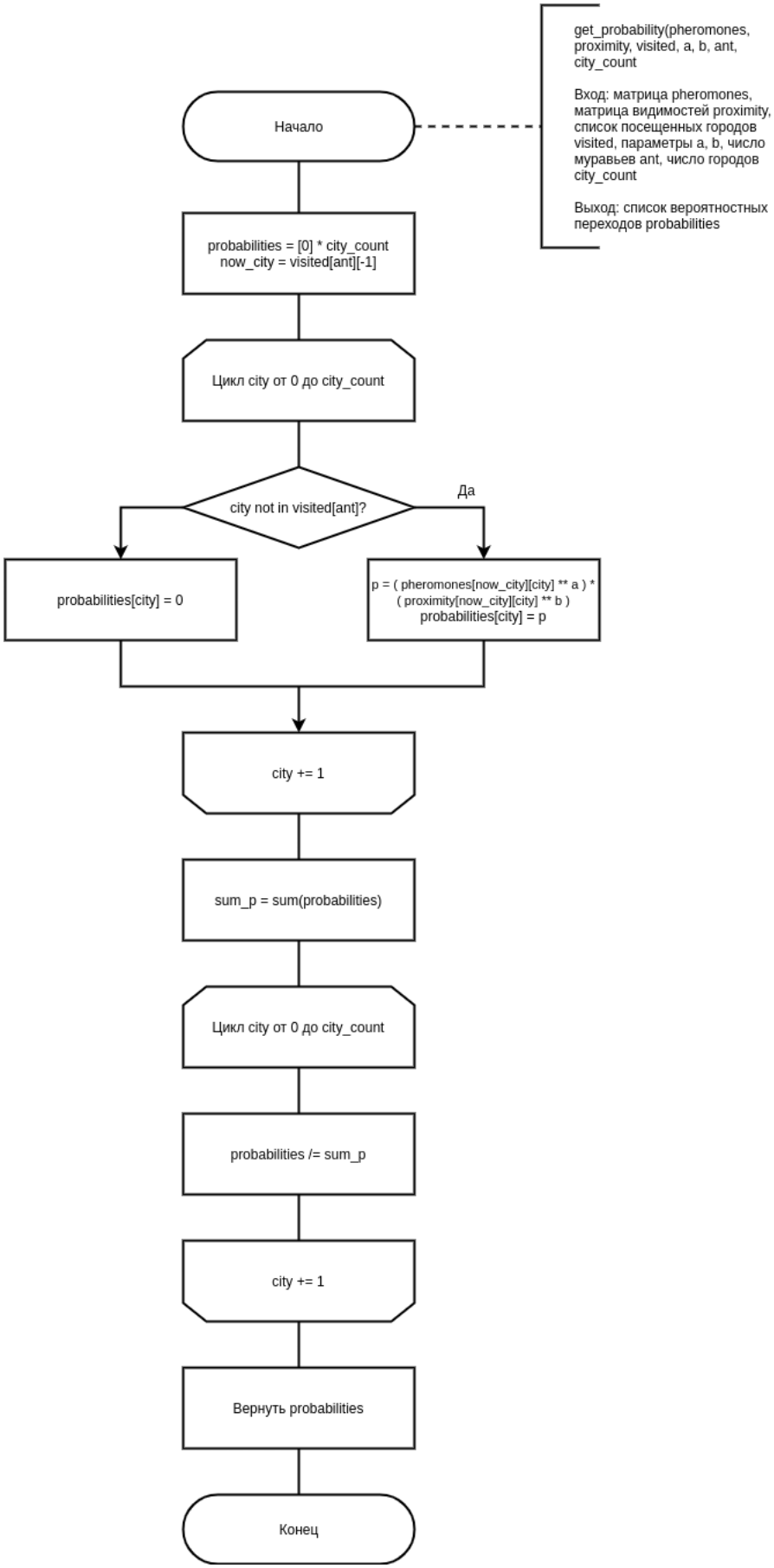

Рисунок 4 – Алгоритм вычисления списка вероятностных переходов

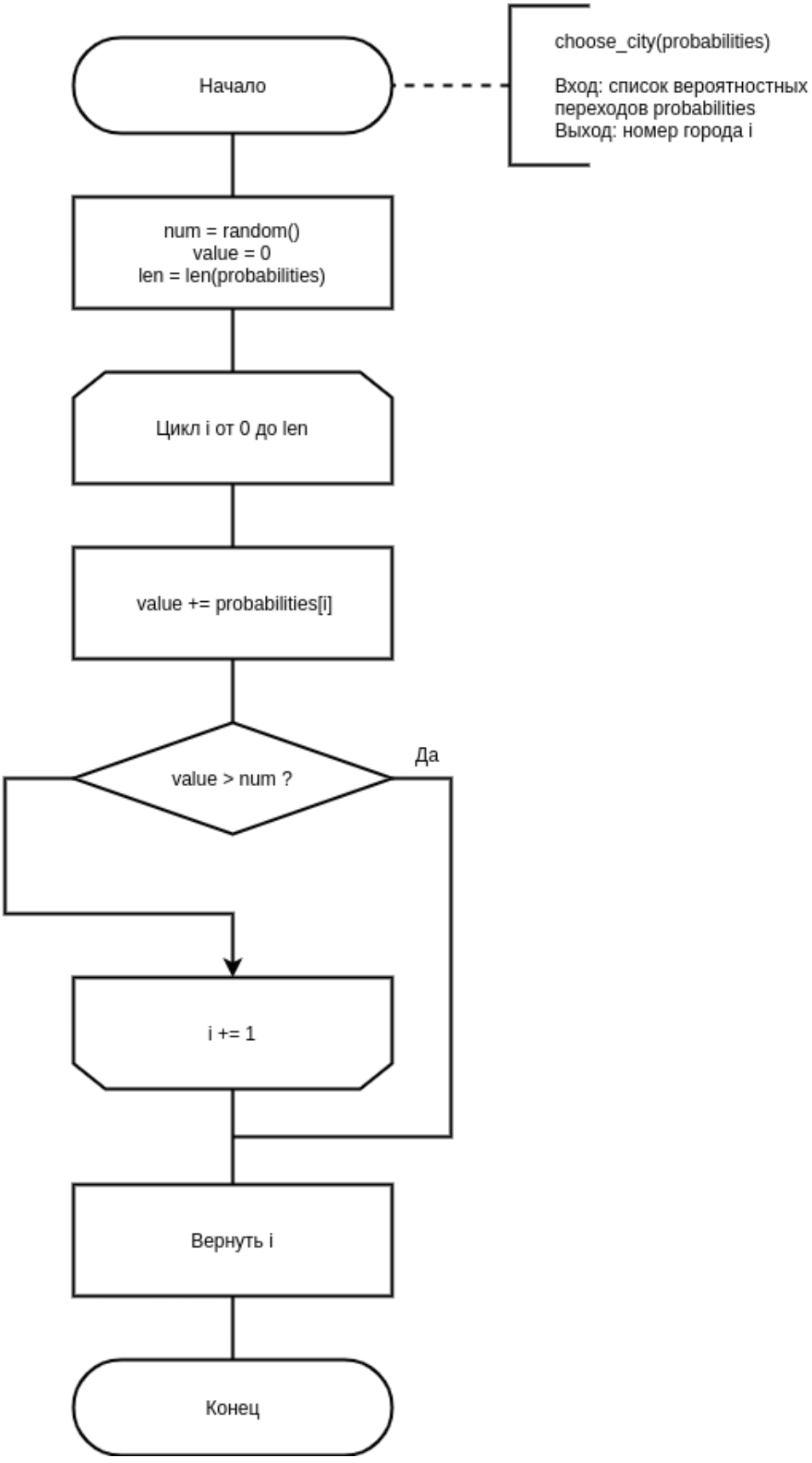

Рисунок 5 – Алгоритм выбора следующего города случайным образом

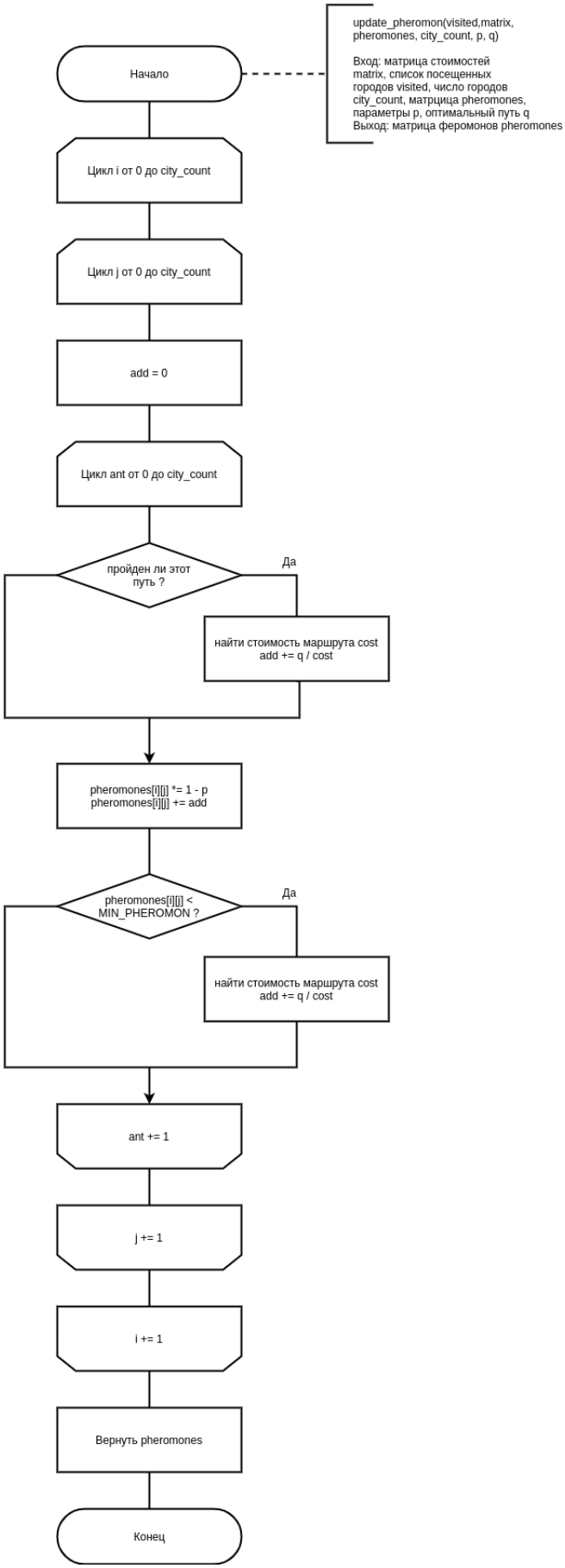

Рисунок 6 – Алгоритм обновления феромона

3 Технологическая часть

3.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

— стоимости путей должны быть целыми числами;

— число городов должно быть больше 1;

— число дней должно быть больше 0;

— параметры муравьиного алгоритма должны быть вещественными неотрицательными числами;

— матрица должна задавать неориентированный граф;

— должно быть выдано сообщение об ошибке при некорректном вводе параметров.

3.2 Средства реализации

В качестве языка программирования для реализации лабораторной работы был выбран Python [3]. Данный выбор обусловлен наличием методов для замера процессорного времени.

Время работы реализованных алгоритмов было замерено с помощью библиотеки time [4].

3.3 Реализация алгоритмов

В листинге 1 приведена реализация алгоритма полного перебора.

Листинг 1 – Реализация алгоритма полного перебора

```
def get_cost(matrix, route):
    now_cost = 0

    for num in range(len(route) - 1):
        start_city = route[num]
        end_city = route[num + 1]

        now_cost += matrix[start_city][end_city]
```

```
9
10      return now_cost
11
12
13  def get_all_routes(cities):
14      routes = []
15
16      for route in itertools.permutations(cities, len(cities)):
17          route = list(route)
18          route.append(route[0])
19          routes.append(route)
20
21      return routes
22
23
24  def brute_force(count, matrix):
25      cities = [city for city in range(count)]
26      routes = get_all_routes(cities)
27
28      min_cost = inf
29
30      for route in routes:
31          now_cost = get_cost(matrix, route)
32
33          if now_cost < min_cost:
34              min_cost = now_cost
35              min_rout = route
36
37      return min_rout, min_cost
```

В листинге 2 приведена реализация муравьиного алгоритма.

Листинг 2 – Реализация муравьиного алгоритма

```
def ant_algorithm(matrix, size, a, b, p, count_days):
    min_cost = inf
    q = get_q(matrix, size)
    proximity = get_proximity(matrix, size)
    pheromones = [[START_PHEROMON
    for i in range(size)] for j in range(size)]
    count_ants = size

    for _ in range(count_days):
        cities = [city for city in range(size)]
        visited = get_visited_cities(cities, count_ants)

        for ant in range(count_ants):
            while len(visited[ant]) != count_ants:
                pk = get_probability(pheromones, proximity,
                                     visited, a, b, ant, size)
                next_city = choose_city(pk)
                visited[ant].append(next_city)
            visited[ant].append(visited[ant][0])

            now_cost = get_cost(matrix, visited[ant])

            if now_cost < min_cost:
                min_cost = now_cost
                min_route = visited[ant]

        pheromones = update_pheromon(visited,
        matrix, pheromones, size, p, q)

    return min_route, min_cost
```

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялся замерный эксперимент:

— операционная система macOS Ventura 13.0.1 [5];

— память 32 ГБ;

— процессор 2,3 ГГц 4-ядерный процессор Intel Core i7.

Замеры проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, окружением, а также непосредственно замерным экспериментом.

4.2 Пример работы программы

На рисунке 7 представлен пример работы программы. Вводится количество кард, далее выводится лог при параллельной обработки и последовательной.

```
МЕНЮ:
1. Полный перебор
2. Муравьиный алгоритм
3. Оба алгоритма
4. Построить графики
5. Замерить время
6. Параметризация
0. Выход
Выбор: 3

Введите число городов: 6

Выберите способ ввода матрицы стоимостей:
    1. Ручной ввод
    2. Случайная матрица
    3. Загрузить из файла
Выбор: 2

Введите нижнюю границу стоимостей: 1
Введите верхнюю границу стоимостей: 9

Исходная матрица стоимостей:
0 8 5 4 9 4
8 0 9 6 7 1
5 9 0 6 9 6
4 6 6 0 4 1
9 7 9 4 0 1
4 1 6 1 1 0


=== Полный перебор ===
Самый короткий маршрут:
1 -> 3 -> 2 -> 6 -> 5 -> 4 -> 1
Его стоимость :  24

Введите параметр α: 0.5
Введите коэффициент испарения: 0.3
Введите число дней: 200


=== Муравьиный алгоритм ===
Самый короткий маршрут:
1 -> 4 -> 5 -> 6 -> 2 -> 3 -> 1
Его стоимость :  24
```

Рисунок 7 – Пример работы программы

4.3 Время выполнения реализованных алгоритмов

Функция process_time из библиотеки time возвращает процессорное время в секундах – значение типа float.

Для замера времени необходимо получить значение времени до начала

выполнения алгоритма, затем после её окончания. Чтобы получить результат, необходимо вычесть из второго значения первое.

Замеры проводились для матриц стоимостей, заполненных случайным образом, размером от 2 до 10. Результаты измерения времени приведены в таблице 1 (в секундах), а их графическое представление - на рисунке 8.

Таблица 1 – Результаты замеров времени

Число городов	Полный перебор	Муравьиный алгоритм
2	0.000327	0.010621
3	0.000039	0.015454
4	0.000121	0.031155
5	0.000706	0.049366
6	0.003409	0.078147
7	0.024055	0.106820
8	0.223502	0.149278
9	2.248946	0.227690
10	22.530937	0.299644

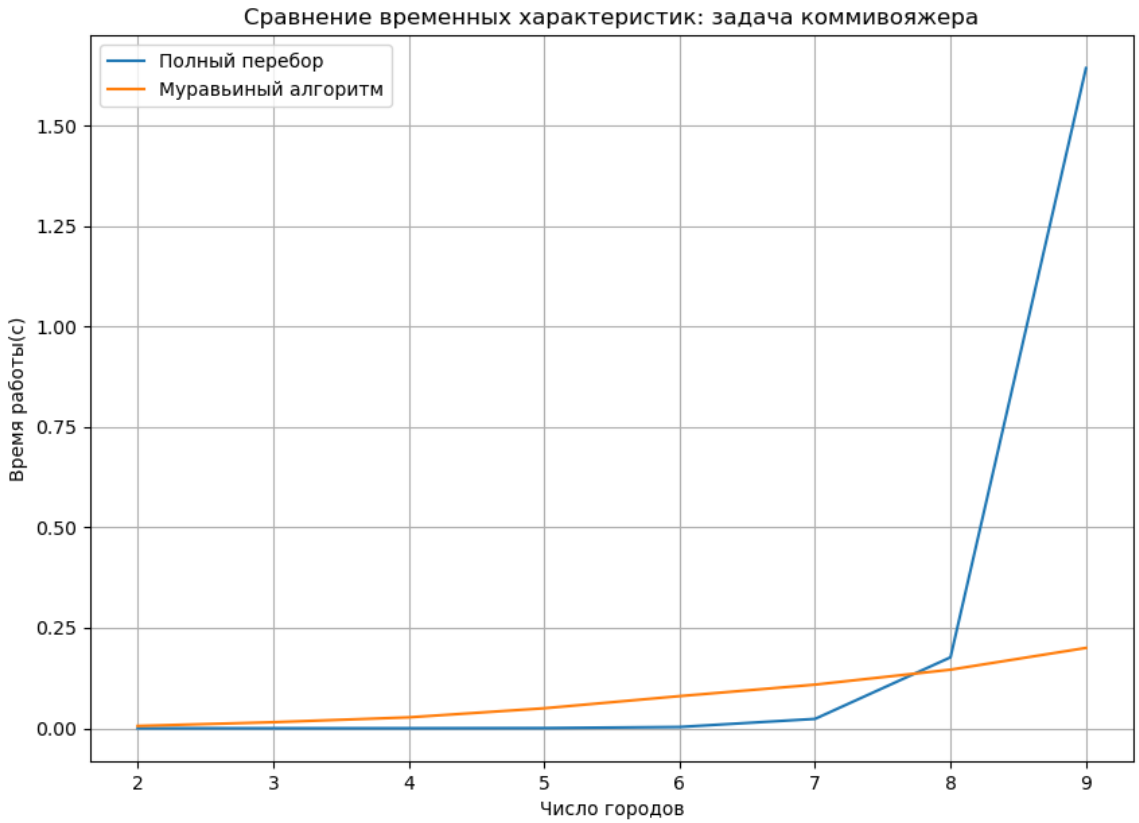

Рисунок 8 – Сравнение по времени алгоритмов задачи коммивояжёра

4.4 Параметризация

Целью проведения параметризации является определение таких комбинаций параметров, при которых муравьиный алгоритм даёт наилучшие результаты.

В результате автоматической параметризации будет получена таблицы со следующими столбцами:

— коэффициент видимости α – изменяющийся параметр;

— коэффициент испарения феромона ρ – изменяющийся параметр;

— число дней $days$ – изменяющийся параметр;

— эталонный результат $ideal$;

— разность полученного при данных параметрах значения и эталонного $mistake$.

4.4.1 Класс данных 1

Класс данных 1 представляет собой матрицу стоимостей в диапазоне от 1 до 5 для 8 городов:

$$K_1 = \begin{pmatrix} 0 & 4 & 3 & 2 & 4 & 5 & 5 & 3 \\ 4 & 0 & 4 & 5 & 5 & 1 & 4 & 5 \\ 3 & 4 & 0 & 5 & 4 & 1 & 2 & 1 \\ 2 & 5 & 5 & 0 & 1 & 3 & 1 & 5 \\ 4 & 5 & 4 & 1 & 0 & 2 & 5 & 4 \\ 5 & 1 & 1 & 3 & 2 & 0 & 4 & 5 \\ 5 & 4 & 2 & 1 & 5 & 4 & 0 & 2 \\ 3 & 5 & 1 & 5 & 4 & 5 & 2 & 0 \end{pmatrix} \tag{6}$$

Для данного класса данных приведена таблица с выборкой параметров, которые наилучшим образом решают поставленную задачу.

Таблица 2 – Параметры для класса данных 1

α	ρ	Дни	Результат	Ошибки
0.1	0.9	100	15	0
0.1	0.9	200	15	0
0.1	0.9	300	15	0
0.1	0.9	400	15	0
0.1	0.9	500	15	0
0.2	0.8	100	15	0
0.2	0.8	200	15	0
0.2	0.8	300	15	0
0.2	0.8	400	15	0
0.2	0.8	500	15	0

Продолжение таблицы 2

α	ρ	Дни	Результат	Ошибки
0.3	0.7	100	15	0
0.3	0.7	200	15	0
0.3	0.7	300	15	0
0.3	0.7	400	15	0
0.3	0.7	500	15	0
0.4	0.6	100	15	0
0.4	0.6	200	15	0
0.4	0.6	300	15	0
0.4	0.6	400	15	0
0.4	0.6	500	15	0
0.5	0.5	100	15	0
0.5	0.5	200	15	0
0.5	0.5	300	15	0
0.5	0.5	400	15	0
0.5	0.5	500	15	0
0.6	0.4	100	15	0
0.6	0.4	200	15	0
0.6	0.4	300	15	0
0.6	0.4	400	15	0
0.6	0.4	500	15	0
0.7	0.3	100	15	0
0.7	0.3	200	15	0
0.7	0.3	300	15	0
0.7	0.3	400	15	0
0.7	0.3	500	15	0

Продолжение таблицы 2

α	ρ	Дни	Результат	Ошибки
0.8	0.2	100	15	0
0.8	0.2	200	15	0
0.8	0.2	300	15	0
0.8	0.2	400	15	0
0.8	0.2	500	15	0
0.9	0.1	100	15	0
0.9	0.1	200	15	0
0.9	0.1	300	15	0
0.9	0.1	400	15	0
0.9	0.1	500	15	0

4.4.2 Класс данных 2

Класс данных 2 представляет собой матрицу стоимостей в диапазоне от 5000 до 9000 для 8 городов:

$$K_1 = \begin{pmatrix} 0 & 5466 & 8308 & 8068 & 7284 & 5635 & 6055 & 8129 \\ 5466 & 0 & 8205 & 7384 & 6794 & 6048 & 6174 & 6306 \\ 8308 & 8205 & 0 & 5485 & 7872 & 7981 & 7868 & 6912 \\ 8068 & 7384 & 5485 & 0 & 7002 & 6683 & 7544 & 8278 \\ 7284 & 6794 & 7872 & 7002 & 0 & 5159 & 8240 & 5663 \\ 5635 & 6048 & 7981 & 6683 & 5159 & 0 & 8801 & 8844 \\ 6055 & 6174 & 7868 & 7544 & 8240 & 8801 & 0 & 5493 \\ 8129 & 6306 & 6912 & 8278 & 5663 & 8844 & 5493 & 0 \end{pmatrix} \quad (7)$$

Для данного класса данных приведена таблица с выборкой параметров, которые наилучшим образом решают поставленную задачу.

Таблица 3 – Параметры для класса данных 2

α	ρ	Дни	Результат	Ошибки
0.1	0.7	300	47326	0
0.1	0.7	400	47326	0
0.1	0.7	500	47326	0
0.2	0.5	300	47326	0
0.2	0.5	400	47326	0
0.2	0.5	500	47326	0
0.3	0.3	300	47326	0
0.3	0.3	400	47326	0
0.3	0.3	500	47326	0
0.4	0.1	300	47326	0
0.4	0.1	400	47326	0
0.4	0.1	500	47326	0
0.5	0.6	300	47326	0
0.5	0.6	400	47326	0
0.5	0.6	500	47326	0
0.6	0.8	300	47326	0
0.6	0.8	400	47326	0
0.6	0.8	500	47326	0
0.7	0.2	300	47326	0
0.7	0.2	400	47326	0
0.7	0.2	500	47326	0

Продолжение таблицы 3

α	ρ	Дни	Результат	Ошибки
0.8	0.6	300	47326	0
0.8	0.6	400	47326	0
0.8	0.6	500	47326	0
0.9	0.9	300	47326	0
0.9	0.9	400	47326	0
0.9	0.9	500	47326	0

Вывод

В результате эксперимента было получено, что для 2 городов полный перебор работает быстрее муравьиного алгоритма в 32 раза. Начиная с 8 городов, муравьиный алгоритм работает быстрее полного перебора: в 75 раз быстрее для 10 городов. Таким образом, муравьиный алгоритм необходимо использовать при большом числе городов - от 8 и более.

Также в результате проведения параметризации было установлено, что для первого класса данных лучшие результаты муравьиный алгоритм дает на следующих значениях параметров:

— α = 0.1, ρ = 0.1-0.5, 0.7-0.9;

— α = 0.2, ρ = 0.1-0.7, 0.9;

— α = 0.3, ρ = 0.2-0.6, 0.9;

— α = 0.4, ρ = 0.5-0.9;

— α = 0.5, ρ = 0.1, 0.5-0.9;

— α = 0.7, ρ = 0.4-0.8.

Можно сделать вывод о том, что данные значения параметров следует использовать для матриц стоимостей в диапазоне от 1 до 5.

Для второго класса данных лучшие результаты муравьиный алгоритм дает на следующих значениях параметров:

— $\alpha = 0.1$, $\rho = 0.1, 0.4, 0.7$;

— $\alpha = 0.2$, $\rho = 0.3, 0.7, 0.9$;

— $\alpha = 0.3$, $\rho = 0.2, 0.6, 0.9$;

— $\alpha = 0.4$, $\rho = 0.7, 0.8$;

— $\alpha = 0.5$, $\rho = 0.2, 0.6\text{-}0.9$;

— $\alpha = 0.5$, $\rho = 0.3, 0.7\text{-}0.9$;

— $\alpha = 0.6$, $\rho = 0.2, 0.6\text{-}0.9$;

— $\alpha = 0.8$, $\rho = 0.1, 0.6$;

— $\alpha = 0.9$, $\rho = 0.9$.

Можно сделать вывод о том, что данные значения параметров следует использовать для матриц стоимостей в диапазоне от 5000 до 9000.

Из результатов параметризации видно, что погрешность результата уменьшается при большом числе дней и меньшем значении коэффициента видимости.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы поставленная цель была достигнута. В ходе выполнения лабораторной работы были решены все задачи:

— описаны схемой алгоритмов методы решения задачи коммивояжёра;
— реализованы методы решения задачи коммивояжёра;
— выполнена оценка трудоёмкости описанных алгоритмов;
— проведен сравнительный анализ двух рассмотренных методов решения задачи коммивояжёра;
— был подготовлен отчет по лабораторной работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Эвристический алгоритм классификации и его нейросетевая интерпретация [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/evristicheskiy-algoritm-klassifikatsii-i-ego-neyrosetevaya-interpretatsiya, свободный – (24.12.2022)

2. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях, Штовба С.Д., 2003

3. Welcome to Python [Электронный ресурс]. – Режим доступа: https://www.python.org (дата обращения: 24.12.2022).

4. Time access and conversions [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/library/time (дата обращения: 24.12.2022).

5. Операционная система macOS Ventura [Электронный ресурс]. – Режим доступа: https://www.apple.com/macos/ventura/, свободный – (06.11.2022)