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 коммивояжёра»

Студент ИУ7-54Б (Группа) _____ (Подпись, дата) Разин А. (И. О. Фамилия)

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

2023 г.

СОДЕРЖАНИЕ

Введение

В последние два десятилетия при оптимизации сложных систем исследователи все чаще применяют природные механизмы поиска наилучших решений. Один из таких механизмов — это муравьиные алгоритмы, представляющие собой новый перспективный метод оптимизации, базирующийся на моделировании поведения колонии муравьев [1].

Первый вариант муравьиного алгоритма был предназначен для приближенного решения задачи коммивояжера [2].

Целью данной работы является разработка программного обеспечения, решающего задачу коммивояжера двумя способами: полным перебором и с помощью муравьиного алгоритма. 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:

- неориентированный граф;
- без элитных муравьев;
- незамкнутый маршрут;
- карта перемещения по Африке.

1 Аналитическая часть

В данной части работы будет описана задача коммивояжера, а также будут описаны способы ее решения — метод полного перебора и метод на основе муравьиного алгоритма.

1.1 Задача коммивояжера

Задача коммивояжера занимает особое место в комбинаторной оптимизации и исследовании операций. Исторически она была одной из тех задач, которые послужили толчком для развития этих направлений. В данной задаче рассматривается n городов и матрица попарных расстояний между ними. 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 [3].

1.2 Решение с использованием полного перебора

Суть данного решения состоит в переборе всех возможных вариантов замкнутых путей и в выборе кратчайшего из них. Данное решение имеет оценку в $O(n!)$, что затрудняет его использование в графах с большим количеством вершин.

1.3 Решение с использованием муравьиного алгоритма

При поиске путей до пищи муравьи общаются друг с другом с помощью феромонов. В случае если путь длинный, феромоны испарятся и последующие сородичи предпочтут иной путь с большим числом феромонов, таким образом наибольшее число феромона будет оставлено на кратчайших путях [1].

Муравей имеет следующие органы чувств:

1) зрение — муравей способен определить длину ребра;

2) память — муравей способен запомнить пройденный маршрут;

3) обоняние — муравей способен чуять феромон.

Введем функцию (1.1), 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.

Вероятность перехода из пункта i в пункт j определяется по формуле (1.2).

$$P_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}, \tag{1.2}$$

где:

1. $\tau_{i,j}$ — количество феромонов на ребре от i до j;
2. $\eta_{i,j}$ — привлекательность пути от i до j;
3. α — параметр влияния расстояния;
4. β — параметр влияния феромона.

При $\alpha = 0$ будет выбран ближайший город, что соответствует жадному алгоритму в классической теории оптимизации. Если $\beta = 0$, работает лишь феромонное усиление, что влечет за собой быстрое вырождение маршрутов к одному субоптимальному решению [1].

После завершения пути, ночью, происходит обновление феромона на пройденных путях по формуле (1.3), в случае, если p — коэффициент испарения феромона, N — количество феромонов, Q — некоторая константа порядка длины путей, L_k — длина пути муравья с номером k [1].

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij}, \quad \Delta\tau_{ij} = \sum_{k=1}^{N} \tau_{ij}^{k}, \tag{1.3}$$

где

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, & \text{ребро посещено } k\text{-ым муравьем,} \\ 0, & \text{иначе.} \end{cases} \tag{1.4}$$

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.

Вывод

В данной части работы были рассмотрены идеи, необходимые для разработки и реализации двух рассматриваемых алгоритмов решения задачи коммивояжера: алгоритма полного перебора и муравьиного алгоритма.

2 Конструкторская часть

В данной части работы будут рассмотрены схемы муравьиного алгоритма и алгоритма полного перебора.

2.1 Требования к программному обеспечению

Программа должна предоставлять следующие возможности:

- выбор файла с матрицей расстояний;
- ввод параметров α, β и $days$ для муравьиного алгоритма;
- вывод на экран найденного каждым из алгоритмов кратчайшего пути и его длины;
- измерение времени получения результатов каждого из рассматриваемых алгоритмов.

2.2 Схемы алгоритмов

На рисунке 2.1 представлен алгоритм перебора всех возможных путей. На рисунках 2.2–2.5, представлен муравьиный алгоритм и схемы алгоритмов расчета данных для муравьиного алгоритма соотвественно.

2.3 Оценка трудоемкости

Задача коммивояжера является NP-трудной, и точный переборный алгоритм ее решения имеет факториальную сложность $O(n!)$ [3].

Сложность муравьиного алгоритма равна $O(t_{max} \cdot m \cdot n^2)$, то есть она зависит от времени жизни колонии, количества городов и количества муравьев в колонии [4].

В разработанной реализации количество муравьев равно количеству городов, следовательно трудоемкость муравьиного алгоритма равна $O(t_{max} \cdot n^3)$.

Вывод

В данной части работы были рассмотрены схемы муравьиного алгоритма и алгоритма полного перебора.

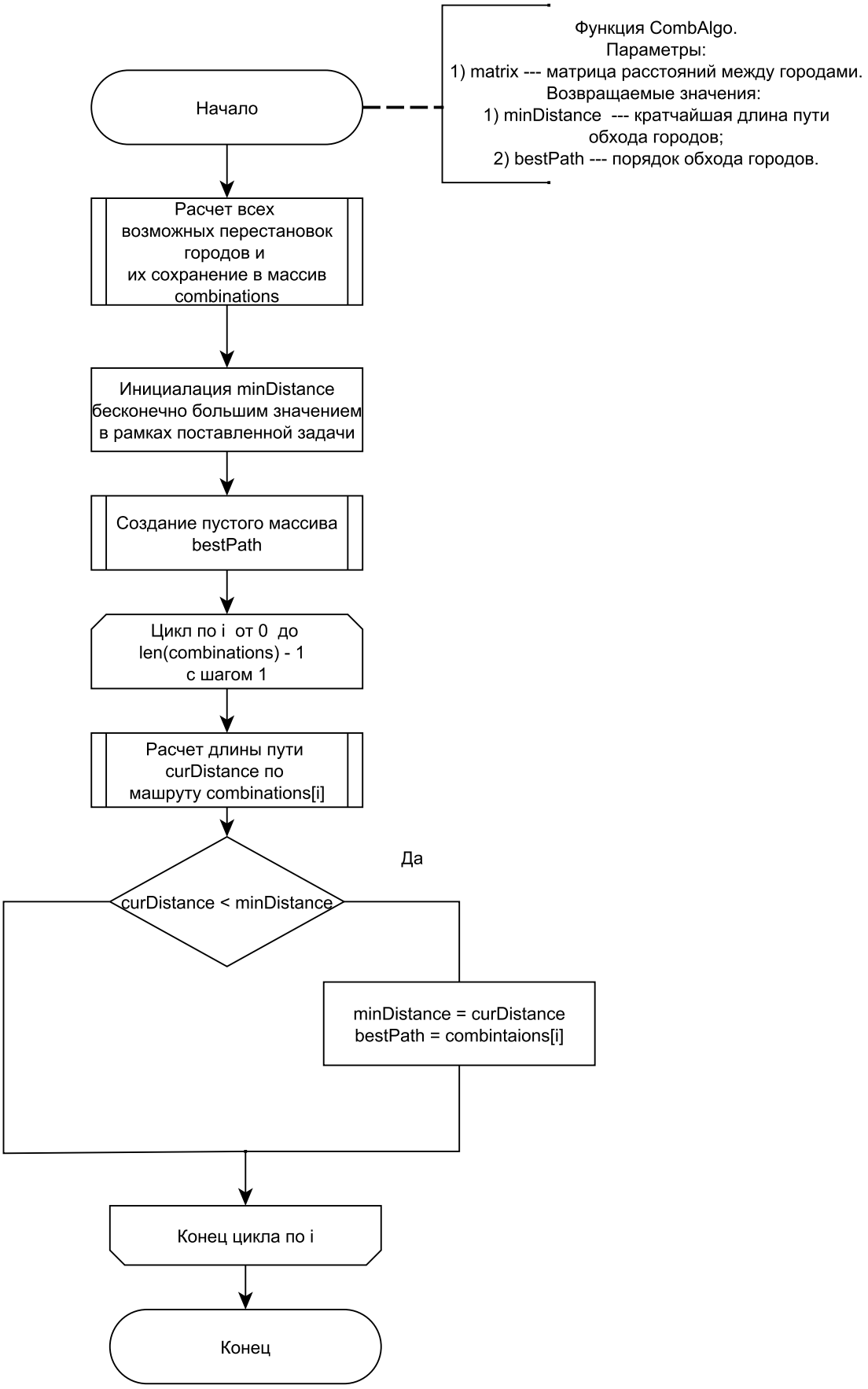

Рисунок 2.1 – Схема алгоритма перебора всех возможных путей

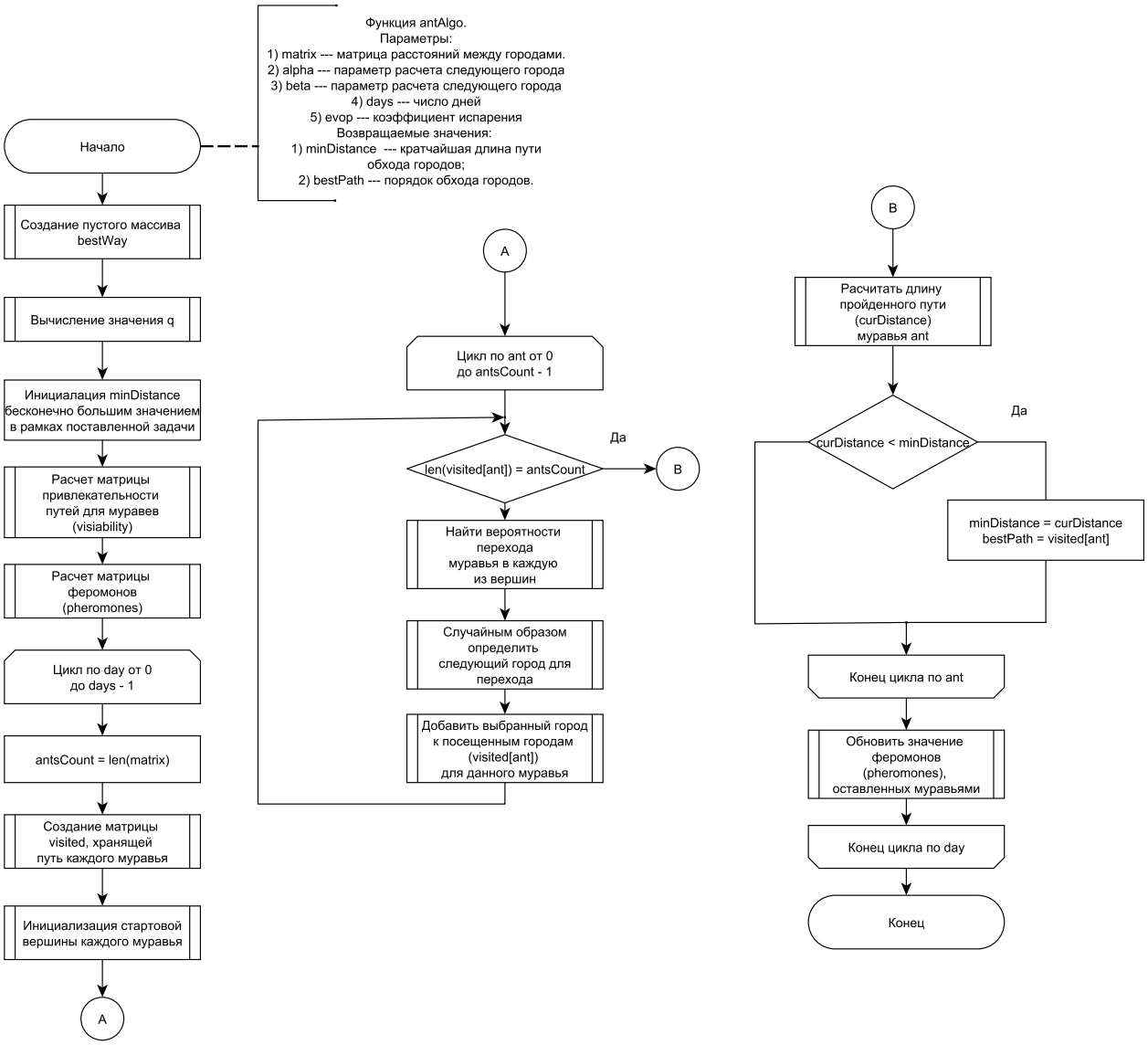

Рисунок 2.2 – Схема муравьиного алгоритма

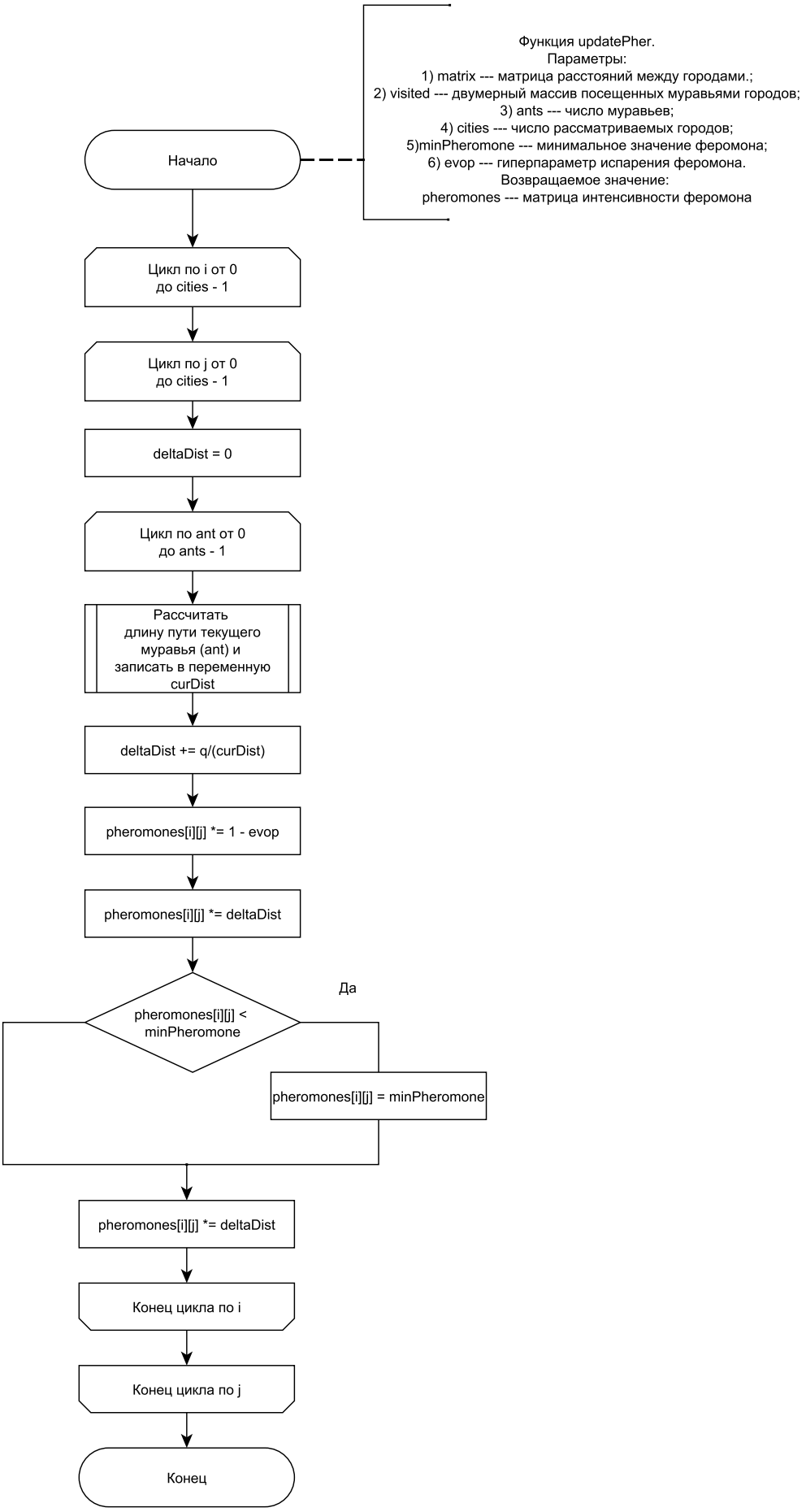

Рисунок 2.3 – Схема алгоритма обновления феромонов

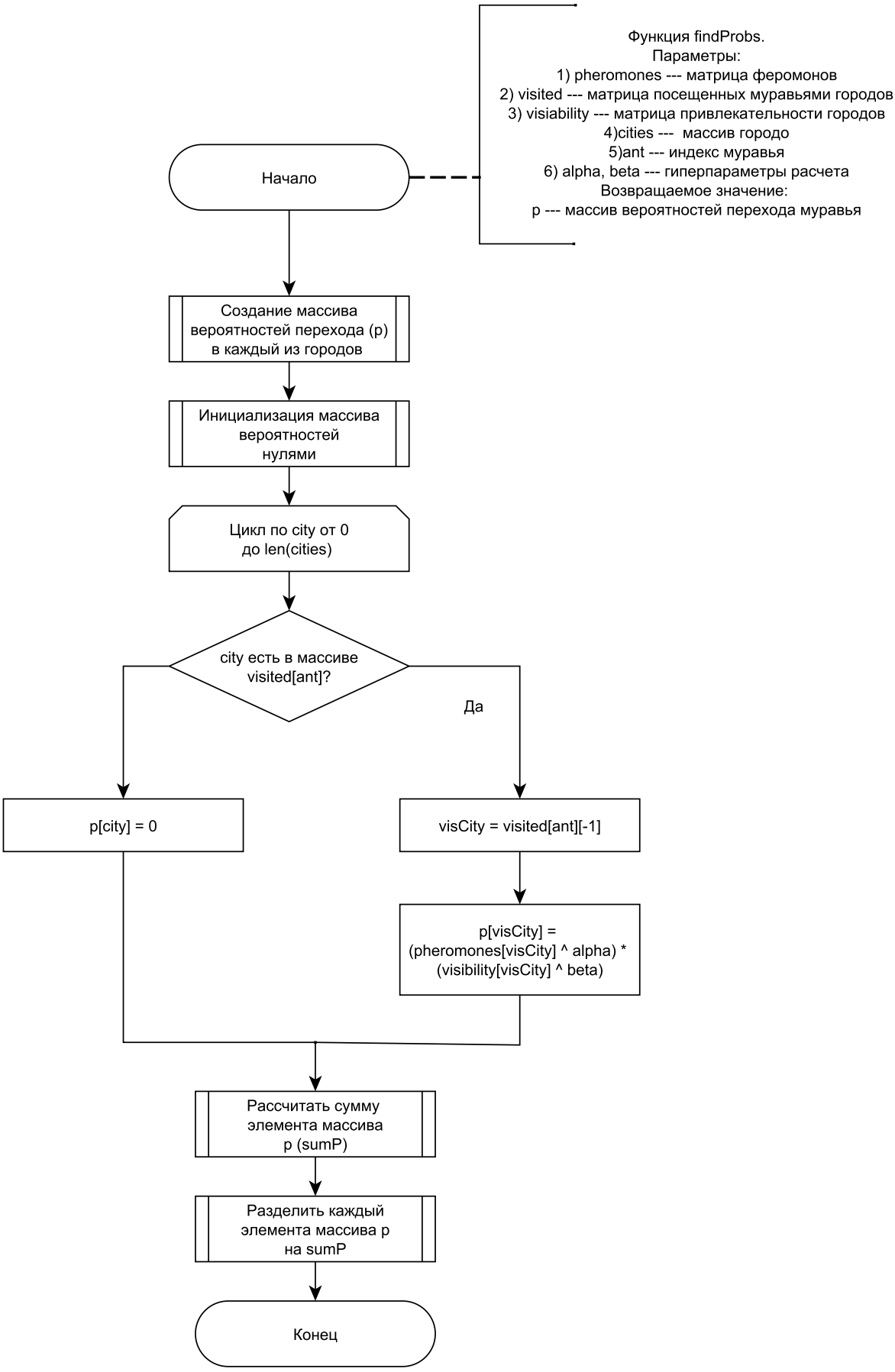

Рисунок 2.4 – Схема алгоритма расчета вероятностей перехода муравья в вершины

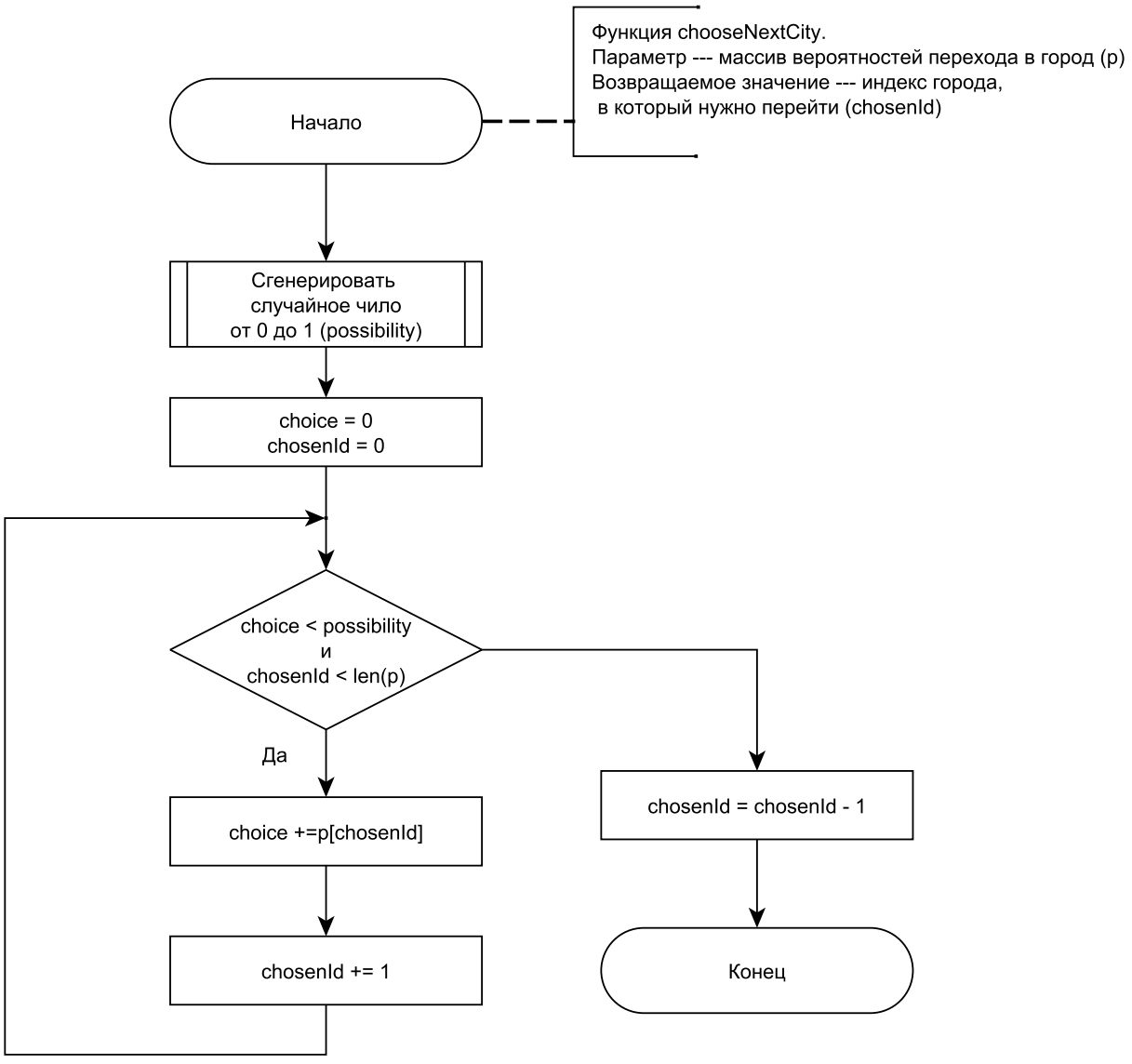

Рисунок 2.5 – Схема алгоритма выбора следующей вершины для перехода

3 Технологический раздел

В данной части работы будут описаны средства реализации программы, а также листинги и функциональные тесты.

3.1 Средства реализации

Алгоритмы для данной лабораторной работы были реализованы на языке `Python`, при использовании компилятора, так как в стандартной библиотеке приведенного языка присутствует функция `process_time`, позволяющая измерять процессорное время [5].

3.2 Реализация алгоритмов

Листинги исходных кодов программ А.1–А.4 приведены в приложении.

3.3 Тестирование

В таблице 3.1, приведены функциональные тесты программы, значения в столбце «Результат программы», обозначает минимальное расстояние и порядок обхода городов, значения в данном столбце были получены при использовании обоих алгоритмов. Все тесты были пройдены успешно.

Таблица 3.1 – Функциональные тесты

Матрица смежности	Ожидаемый результат	Результат программы
$\begin{pmatrix} 0 & 1 & 5 & 6 \\ 1 & 0 & 8 & 8 \\ 5 & 8 & 0 & 10 \\ 6 & 8 & 10 & 0 \end{pmatrix}$	14, [2, 0, 1, 3]	14, [2, 0, 1, 3]
$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{pmatrix}$	3, [1, 0, 2]	3, [1, 0, 2]
$\begin{pmatrix} 0 & 3 \\ 3 & 0 \end{pmatrix}$	3, [0, 1]	3, [0, 1]

Вывод

В данной части работы были описаны средства реализации и представлены листинги реализованных алгоритмов и тесты, успешно пройденные программой.

4 Исследовательская часть

В данном разделе будет описано исследование зависимости среднего числа генерируемых кадров от числа и типа примитивов на сцене. Также будет описаны технические характеристики устройства, на котором проводились замеры и приведен анализ полученных результатов.

4.1 Технические характеристики

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

1. Процессор Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 МГц, ядер: 6, логических процессоров: 12.
2. Оперативная память: 16 ГБайт.
3. Операционная система: Майкрософт Windows 10 Pro [6].
4. Использованная подсистема: WSL2 [7].

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

4.2 Параметризация муравьиного алгоритма

В качестве исходной матрицы для параметаризации была использована матрица В результате параметаризации была получена таблицы Б.1–Б.3, приведенные в приложении, таблицы имеют различный разброс расстояний: 1000, 100, 10 соответственно. В данных таблице столбцы обозначают соответственно:

1. α — параметр α при вычислении вероятности перехода в новый город;
2. eva — коэффициент испарения феромонов;
3. $days$ — время жизни колонии;
4. $optim$ — результат решения полным перебором;
5. $delta$ — разница между решением полным перебором и решением муравьиного алгоритма.

Замеры проводились 10 раз и выбирался результат с максимальным *delta* от результата перебора. Для рассматриваемых таблиц Б.1–Б.3 соответственно использовались матрицы расстояний (4.1–4.3).

$$K_1 = \begin{bmatrix} 0 & 3434 & 934 & 3096 & 5374 & 3486 & 1740 & 1328 & 2672 & 2547 \\ 3434 & 0 & 4315 & 4726 & 3236 & 699 & 5121 & 2290 & 6050 & 5950 \\ 934 & 4315 & 0 & 3468 & 5892 & 4410 & 1161 & 2084 & 1997 & 1831 \\ 3096 & 4726 & 3468 & 0 & 7728 & 4259 & 2888 & 3917 & 3239 & 3330 \\ 5374 & 3236 & 5892 & 7728 & 0 & 3904 & 7021 & 4088 & 7889 & 7717 \\ 3486 & 699 & 4410 & 4259 & 3904 & 0 & 5069 & 2539 & 5974 & 5899 \\ 1740 & 5121 & 1161 & 2888 & 7021 & 5069 & 0 & 3061 & 934 & 831 \\ 1328 & 2290 & 2084 & 3917 & 4088 & 2539 & 3061 & 0 & 3987 & 3851 \\ 2672 & 6050 & 1997 & 3239 & 7889 & 5974 & 934 & 3987 & 0 & 205 \\ 2547 & 5950 & 1831 & 3330 & 7717 & 5899 & 831 & 3851 & 205 & 0 \end{bmatrix} \tag{4.1}$$

$$K_2 = \begin{bmatrix} 0 & 48 & 72 & 36 & 34 & 89 & 85 & 8 & 22 & 11 \\ 48 & 0 & 33 & 23 & 78 & 61 & 16 & 1 & 92 & 31 \\ 72 & 33 & 0 & 100 & 51 & 9 & 66 & 58 & 36 & 23 \\ 36 & 23 & 100 & 0 & 41 & 43 & 45 & 51 & 31 & 67 \\ 34 & 78 & 51 & 41 & 0 & 33 & 22 & 38 & 32 & 63 \\ 89 & 61 & 9 & 43 & 33 & 0 & 46 & 71 & 9 & 32 \\ 85 & 16 & 66 & 45 & 22 & 46 & 0 & 41 & 51 & 78 \\ 8 & 1 & 58 & 51 & 38 & 71 & 41 & 0 & 100 & 23 \\ 22 & 92 & 36 & 31 & 32 & 9 & 51 & 100 & 0 & 33 \\ 11 & 31 & 23 & 67 & 63 & 32 & 78 & 23 & 33 & 0 \end{bmatrix} \tag{4.2}$$

$$K_3 = \begin{bmatrix} 0 & 1 & 9 & 0 & 9 & 5 & 2 & 10 & 7 & 10 \\ 1 & 0 & 9 & 10 & 9 & 1 & 9 & 1 & 9 & 7 \\ 9 & 9 & 0 & 4 & 8 & 7 & 4 & 6 & 8 & 5 \\ 0 & 10 & 4 & 0 & 9 & 0 & 1 & 7 & 3 & 1 \\ 9 & 9 & 8 & 9 & 0 & 8 & 9 & 8 & 7 & 5 \\ 5 & 1 & 7 & 0 & 8 & 0 & 1 & 4 & 8 & 2 \\ 2 & 9 & 4 & 1 & 9 & 1 & 0 & 9 & 8 & 9 \\ 10 & 1 & 6 & 7 & 8 & 4 & 9 & 0 & 4 & 10 \\ 7 & 9 & 8 & 3 & 7 & 8 & 8 & 4 & 0 & 9 \\ 10 & 7 & 5 & 1 & 5 & 2 & 9 & 10 & 9 & 0 \end{bmatrix} \tag{4.3}$$

4.3 Проведение замеров

В результате замеров времени была получена таблица 4.1, замеры проводились 10 раз, после чего время усреднялось, значение столбца n представляет собой размерность матрицы расстояний, иные 2 столбца представляют время получения результата реализаций соответствующих алгоритмов в секундах. По таблице 4.1 был построен график 4.1.

Таблица 4.1 – Полученная таблица замеров по времени времени работы реализаций муравьиного алгоритма и алгоритма полного перебора

n	Алгоритм полного перебора (с)	Муравьиный алгоритм (с)
2	0.000068	0.003853
3	0.000042	0.011109
4	0.000080	0.027333
5	0.000256	0.060201
6	0.001455	0.101121
7	0.011381	0.178127
8	0.099935	0.318359
9	1.154073	0.476853
10	13.011171	0.646156

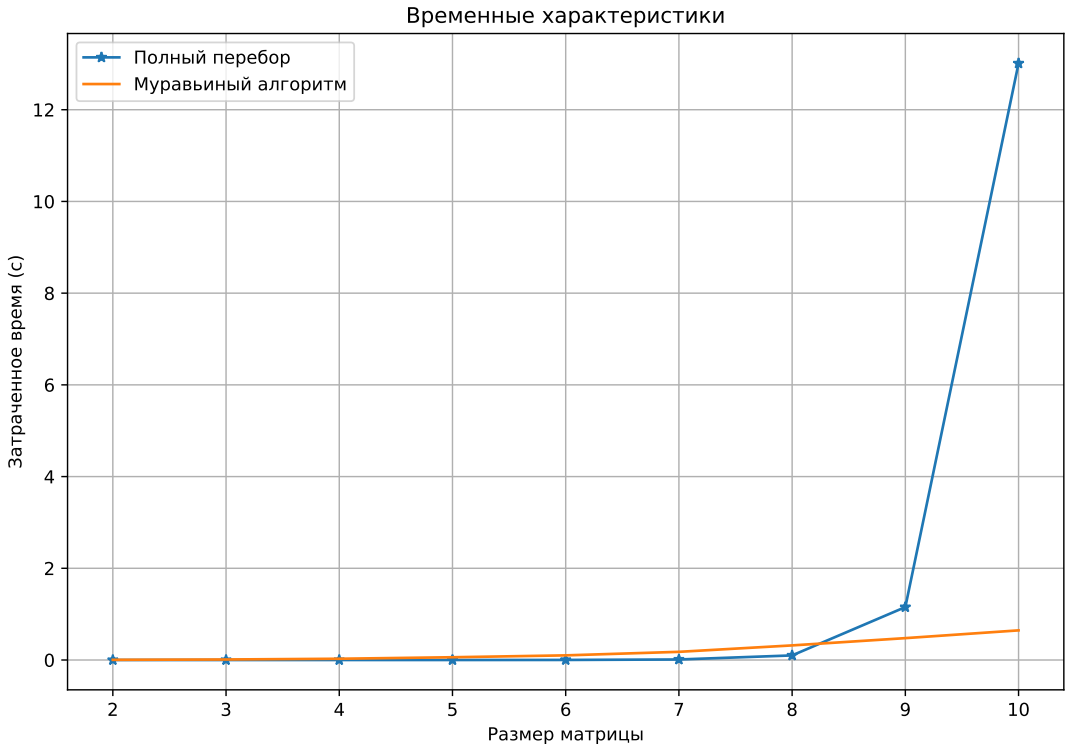

Рисунок 4.1 – Зависимость скорости получения результата от размерности матрицы

Выводы

Из таблиц Б.1–Б.3 можно сделать вывод, что при увлечении числа дней существования колонии *delta* уменьшается, независимо от разброса расстояний. От значений α и *eva* зависимостей не обнаружено, что говорит о том, что значения параметров должны регулироваться для каждой конкретной задачи отдельно.

Из таблицы 4.1 можно сделать вывод, что при малых размерностях матриц реализация алгоритма полного перебора тратит меньше времени на получение результата. При размерности матрицы 8, реализации алгоритма полного перебора, необходимо 0.09 секунд для получения результата, что в 3.13 меньше времени при сравнении с муравьиным алгоритмом. Начиная с размерности 9, реализации алгоритма полного перебора необходимо 1.15 секунд, что в 1.45 раз больше, чем при использовании муравьиного алгоритма. Таким образом, выбор алгоритма зависит от размера матрицы расстояний, введенной в конкретной задаче.

ЗАКЛЮЧЕНИЕ

В результате исследования можно сделать вывод, что при малых размерностях матриц реализация алгоритма полного перебора тратит меньше времени на получение результата. При размерности матрицы 8, реализации алгоритма полного перебора, необходимо 0.09 секунд для получения результата, что в 3.13 меньше времени при сравнении с муравьиным алгоритмом. Начиная с размерности 9, реализации алгоритма полного перебора необходимо 1.15 секунд, что в 1.45 раз больше, чем при использовании муравьиного алгоритма. Таким образом, выбор алгоритма зависит от размера матрицы расстояний, введенной в конкретной задаче.

При увлечении числа дней существования колонии отличия результата муравьиного алгоритма от оптимального уменьшается, независимо от разброса расстояний. От значений α и eva зависимостей не обнаружено, что говорит о том, что значения параметров должны регулироваться для каждой конкретной задачи отдельно.

Поставленная цель: разработка программного обеспечения, решающего задачу полным перебором и с помощью муравьиного алгоритма, была достигнута.

Для поставленной цели были выполнены все 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.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Штовба, С. Д. Муравьиные алгоритмы // Exponenta Pro. — 2003. — № 4. — С. 70–75.
2. Ершов, Н. М. Лекция 10. Муравьиные алгоритмы // ВМК МГУ. — 2011. — С. 1–14.
3. Задача коммивояжера [Электронный ресурс]. — Режим доступа: http://old.math.nsc.ru/LBRT/k5/OR-MMF/TSPr.pdf (дата обращения: 28.11.2022).
4. Ульянов, М. В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ // М.: Наука-Физматлит. — 2007. — С. 201—207.
5. time — Time access and conversions [Электронный ресурс]. — Режим доступа: https://docs.python.org/3/library/time.html (дата обращения: 22.09.2022).
6. Windows client documentation for IT Pros [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/windows/resources/ (дата обращения: 22.09.2022).
7. Что такое WSL [Электронный ресурс]. — — Режим доступа: https://learn.microsoft.com/ru-ru/windows/wsl/about (дата обращения: 28.09.2023).

ПРИЛОЖЕНИЕ А

Листинг А.1 – Реализация муравьиного алгоритма

```
def antAlg(matrix, cities, alpha, beta, k_evaporation, days):
    q = calcQ(matrix, cities)
    bestWay = []
    minDist = float("inf")
    pheromones = calcPheromones(cities)
    visibility = calcVisibility(matrix, cities)
    ants = cities
    for day in range(days):
        route = np.arange(cities)
        visited = calcVisitedPlaces(route, ants)

        for ant in range(ants):
            while (len(visited[ant]) != ants):
                pk = findProbs(pheromones, visibility, visited,
                   cities, ant, alpha, beta)
                chosenPlace = chooseNextCity(pk)
                visited[ant].append(chosenPlace - 1)

            curLength = calcLength(matrix, visited[ant])

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

        pheromones = updPher(matrix, cities, visited,
           pheromones, q, k_evaporation)

    return minDist, bestWay
```

Листинг А.2 – Реализация алгоритма полного перебора

```
def CombAlg(matrix, size):

    places = np.arange(size)
    placesCombinations = list()

    for combination in it.permutations(places):
        combArr = list(combination)
```

```
        placesCombinations.append(combArr)

    minDist = float("inf")

    for i in range(len(placesCombinations)):
        curDist = 0
        for j in range(size - 1):
            startCity = placesCombinations[i][j]
            endCity = placesCombinations[i][j + 1]
            curDist += matrix[startCity][endCity]

        if (curDist < minDist):
            minDist = curDist
            bestWay = placesCombinations[i]

    return minDist, bestWay
```

Листинг А.3 – Реализация алгоритма расчета вероятностей перехода в не посещенную вершину графа

```
def findProbs(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 А.4 – Реализация алгоритма расчета значения феромонов

```
def updPher(matrix, cities, visited, pheromones, q,
   k_evaporation):
```

```
    ants = cities

    for i in range(cities):
        for j in range(cities):
            delta = 0
            for ant in range(ants):
                cur_dist = calcLength(matrix, visited[ant])
                delta += q / cur_dist

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

    return pheromones
```

ПРИЛОЖЕНИЕ Б

Таблица Б.1 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 1000

α	eva	$days$	$optim$	$delta$
0.1	0.1	1	14175	9659
0.1	0.1	5	14175	5136
0.1	0.1	10	14175	5134
0.1	0.1	100	14175	2520
0.1	0.3	1	14175	9035
0.1	0.3	5	14175	6456
0.1	0.3	10	14175	4638
0.1	0.3	100	14175	2589
0.1	0.5	1	14175	9211
0.1	0.5	5	14175	7446
0.1	0.5	10	14175	5054
0.1	0.5	100	14175	2447
0.1	0.7	1	14175	9498
0.1	0.7	5	14175	7173
0.1	0.7	10	14175	5195
0.1	0.7	100	14175	1916
0.1	0.9	1	14175	9798
0.1	0.9	5	14175	5931
0.1	0.9	10	14175	4353
0.1	0.9	100	14175	2398
0.3	0.1	1	14175	9204
0.3	0.1	5	14175	5735
0.3	0.1	10	14175	6575
0.3	0.1	100	14175	2702
0.3	0.3	1	14175	9728
0.3	0.3	5	14175	6370
0.3	0.3	10	14175	5262

Таблица Б.1 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 1000

α	eva	$days$	$optim$	$delta$
0.3	0.3	100	14175	4026
0.3	0.5	1	14175	8919
0.3	0.5	5	14175	6711
0.3	0.5	10	14175	5348
0.3	0.5	100	14175	3687
0.3	0.7	1	14175	9995
0.3	0.7	5	14175	6243
0.3	0.7	10	14175	5778
0.3	0.7	100	14175	3136
0.3	0.9	1	14175	10299
0.3	0.9	5	14175	6384
0.3	0.9	10	14175	6103
0.3	0.9	100	14175	2810
0.5	0.1	1	14175	13312
0.5	0.1	5	14175	7872
0.5	0.1	10	14175	7133
0.5	0.1	100	14175	3249
0.5	0.3	1	14175	13096
0.5	0.3	5	14175	9309
0.5	0.3	10	14175	7195
0.5	0.3	100	14175	4199
0.5	0.5	1	14175	13583
0.5	0.5	5	14175	7812
0.5	0.5	10	14175	6121
0.5	0.5	100	14175	2845
0.5	0.7	1	14175	12834
0.5	0.7	5	14175	7869
0.5	0.7	10	14175	5688
0.5	0.7	100	14175	4022
0.5	0.9	1	14175	10664

Таблица Б.1 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 1000

α	eva	$days$	$optim$	$delta$
0.5	0.9	5	14175	7691
0.5	0.9	10	14175	7180
0.5	0.9	100	14175	3505
0.7	0.1	1	14175	11347
0.7	0.1	5	14175	7896
0.7	0.1	10	14175	8438
0.7	0.1	100	14175	4893
0.7	0.3	1	14175	15177
0.7	0.3	5	14175	9030
0.7	0.3	10	14175	8237
0.7	0.3	100	14175	4836
0.7	0.5	1	14175	12447
0.7	0.5	5	14175	10137
0.7	0.5	10	14175	7519
0.7	0.5	100	14175	5392
0.7	0.7	1	14175	12591
0.7	0.7	5	14175	9034
0.7	0.7	10	14175	7801
0.7	0.7	100	14175	4708
0.7	0.9	1	14175	11891
0.7	0.9	5	14175	8666
0.7	0.9	10	14175	7681
0.7	0.9	100	14175	3694
0.9	0.1	1	14175	16039
0.9	0.1	5	14175	9178
0.9	0.1	10	14175	9883
0.9	0.1	100	14175	5010
0.9	0.3	1	14175	14475
0.9	0.3	5	14175	11579
0.9	0.3	10	14175	9917

Таблица Б.1 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 1000

α	eva	$days$	$optim$	$delta$
0.9	0.3	100	14175	5361
0.9	0.5	1	14175	16168
0.9	0.5	5	14175	9569
0.9	0.5	10	14175	8310
0.9	0.5	100	14175	6191
0.9	0.7	1	14175	12815
0.9	0.7	5	14175	11116
0.9	0.7	10	14175	10824
0.9	0.7	100	14175	5996
0.9	0.9	1	14175	18027
0.9	0.9	5	14175	11755
0.9	0.9	10	14175	8604
0.9	0.9	100	14175	5834

Таблица Б.2 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 100

α	eva	$days$	$optim$	$delta$
0.1	0.1	5	130	63
0.1	0.1	10	130	77
0.1	0.1	100	130	21
0.1	0.3	1	130	123
0.1	0.3	5	130	92
0.1	0.3	10	130	58
0.1	0.3	100	130	20
0.1	0.5	1	130	122
0.1	0.5	5	130	75
0.1	0.5	10	130	59
0.1	0.5	100	130	24
0.1	0.7	1	130	116
0.1	0.7	5	130	70

Таблица Б.2 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 100

α	eva	$days$	$optim$	$delta$
0.1	0.7	10	130	48
0.1	0.7	100	130	21
0.1	0.9	1	130	144
0.1	0.9	5	130	79
0.1	0.9	10	130	57
0.1	0.9	100	130	26
0.3	0.1	1	130	121
0.3	0.1	5	130	70
0.3	0.1	10	130	58
0.3	0.1	100	130	30
0.3	0.3	1	130	144
0.3	0.3	5	130	85
0.3	0.3	10	130	72
0.3	0.3	100	130	36
0.3	0.5	1	130	126
0.3	0.5	5	130	72
0.3	0.5	10	130	66
0.3	0.5	100	130	38
0.3	0.7	1	130	120
0.3	0.7	5	130	82
0.3	0.7	10	130	83
0.3	0.7	100	130	31
0.3	0.9	1	130	161
0.3	0.9	5	130	99
0.3	0.9	10	130	75
0.3	0.9	100	130	35
0.5	0.1	1	130	182
0.5	0.1	5	130	112
0.5	0.1	10	130	87
0.5	0.1	100	130	36

Таблица Б.2 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 100

α	eva	$days$	$optim$	$delta$
0.5	0.3	1	130	168
0.5	0.3	5	130	101
0.5	0.3	10	130	75
0.5	0.3	100	130	45
0.5	0.5	1	130	151
0.5	0.5	5	130	99
0.5	0.5	10	130	68
0.5	0.5	100	130	50
0.5	0.7	1	130	128
0.5	0.7	5	130	113
0.5	0.7	10	130	83
0.5	0.7	100	130	60
0.5	0.9	1	130	170
0.5	0.9	5	130	108
0.5	0.9	10	130	84
0.5	0.9	100	130	45
0.7	0.1	1	130	164
0.7	0.1	5	130	127
0.7	0.1	10	130	115
0.7	0.1	100	130	67
0.7	0.3	1	130	170
0.7	0.3	5	130	116
0.7	0.3	10	130	92
0.7	0.3	100	130	52
0.7	0.5	1	130	185
0.7	0.5	5	130	113
0.7	0.5	10	130	109
0.7	0.5	100	130	70
0.7	0.7	1	130	185
0.7	0.7	5	130	115

Таблица Б.2 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 100

α	eva	$days$	$optim$	$delta$
0.7	0.7	10	130	98
0.7	0.7	100	130	64
0.7	0.9	1	130	197
0.7	0.9	5	130	143
0.7	0.9	10	130	104
0.7	0.9	100	130	71
0.9	0.1	1	130	215
0.9	0.1	5	130	141
0.9	0.1	10	130	136
0.9	0.1	100	130	80
0.9	0.3	1	130	179
0.9	0.3	5	130	145
0.9	0.3	10	130	141
0.9	0.3	100	130	75
0.9	0.5	1	130	181
0.9	0.5	5	130	152
0.9	0.5	10	130	121
0.9	0.5	100	130	100
0.9	0.7	1	130	208
0.9	0.7	5	130	135
0.9	0.7	10	130	129
0.9	0.7	100	130	73
0.9	0.9	1	130	225
0.9	0.9	5	130	150
0.9	0.9	10	130	137
0.9	0.9	100	130	85

Таблица Б.3 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 10

α	eva	$days$	$optim$	$delta$
0.1	0.1	5	20	7
0.1	0.1	10	20	5
0.1	0.1	100	20	2
0.1	0.3	1	20	12
0.1	0.3	5	20	6
0.1	0.3	10	20	5
0.1	0.3	100	20	2
0.1	0.5	1	20	13
0.1	0.5	5	20	6
0.1	0.5	10	20	6
0.1	0.5	100	20	1
0.1	0.7	1	20	16
0.1	0.7	5	20	6
0.1	0.7	10	20	4
0.1	0.7	100	20	2
0.1	0.9	1	20	14
0.1	0.9	5	20	7
0.1	0.9	10	20	5
0.1	0.9	100	20	1
0.3	0.1	1	20	11
0.3	0.1	5	20	9
0.3	0.1	10	20	6
0.3	0.1	100	20	2
0.3	0.3	1	20	11
0.3	0.3	5	20	9
0.3	0.3	10	20	6
0.3	0.3	100	20	2
0.3	0.5	1	20	11
0.3	0.5	5	20	8
0.3	0.5	10	20	5

Таблица Б.3 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 10

α	eva	$days$	$optim$	$delta$
0.3	0.5	100	20	2
0.3	0.7	1	20	13
0.3	0.7	5	20	6
0.3	0.7	10	20	7
0.3	0.7	100	20	2
0.3	0.9	1	20	12
0.3	0.9	5	20	8
0.3	0.9	10	20	8
0.3	0.9	100	20	1
0.5	0.1	1	20	13
0.5	0.1	5	20	9
0.5	0.1	10	20	8
0.5	0.1	100	20	3
0.5	0.3	1	20	12
0.5	0.3	5	20	7
0.5	0.3	10	20	6
0.5	0.3	100	20	4
0.5	0.5	1	20	13
0.5	0.5	5	20	7
0.5	0.5	10	20	8
0.5	0.5	100	20	3
0.5	0.7	1	20	17
0.5	0.7	5	20	7
0.5	0.7	10	20	5
0.5	0.7	100	20	2
0.5	0.9	1	20	12
0.5	0.9	5	20	9
0.5	0.9	10	20	7
0.5	0.9	100	20	2
0.7	0.1	1	20	13

Таблица Б.3 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 10

α	eva	$days$	$optim$	$delta$
0.7	0.1	5	20	10
0.7	0.1	10	20	7
0.7	0.1	100	20	3
0.7	0.3	1	20	14
0.7	0.3	5	20	8
0.7	0.3	10	20	8
0.7	0.3	100	20	4
0.7	0.5	1	20	14
0.7	0.5	5	20	10
0.7	0.5	10	20	8
0.7	0.5	100	20	4
0.7	0.7	1	20	13
0.7	0.7	5	20	11
0.7	0.7	10	20	7
0.7	0.7	100	20	4
0.7	0.9	1	20	16
0.7	0.9	5	20	10
0.7	0.9	10	20	7
0.7	0.9	100	20	3
0.9	0.1	1	20	17
0.9	0.1	5	20	11
0.9	0.1	10	20	9
0.9	0.1	100	20	4
0.9	0.3	1	20	15
0.9	0.3	5	20	11
0.9	0.3	10	20	9
0.9	0.3	100	20	5
0.9	0.5	1	20	14
0.9	0.5	5	20	11
0.9	0.5	10	20	10

Таблица Б.3 – Полученная таблица параметаризации муравьиного алгоритма с разбросом длин путей 10

α	eva	$days$	$optim$	$delta$
0.9	0.5	100	20	6
0.9	0.7	1	20	15
0.9	0.7	5	20	13
0.9	0.7	10	20	10
0.9	0.7	100	20	5
0.9	0.9	1	20	19
0.9	0.9	5	20	11
0.9	0.9	10	20	11
0.9	0.9	100	20	4