**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Решение задачи о коммивояжере с помощью метода ближайшего соседа»**

**Студент гр. 22Б16-пу \_\_\_\_\_\_\_\_\_\_\_\_\_ Шарабарин М.С.**

**Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

# Оглавление

[Оглавление 2](#_Toc1709610656)

[Цель работы 3](#_Toc124072108)

[Описание задачи (формализация задачи) 3](#_Toc259608225)

[Теоретическая часть 3](#_Toc917857674)

[Основные шаги программы 4](#_Toc281540400)

[Сравнительный анализ: 4](#_Toc1124058329)

[Ограничения: 4](#_Toc494639595)

[Основные шаги программы 5](#_Toc1755642168)

[Блок схема программы 6](#_Toc968683058)

[Описание программы 7](#_Toc107777993)

[Рекомендации для пользователя 7](#_Toc672695833)

[Рекомендации для программиста 7](#_Toc1271854660)

[Исходный код программы 8](#_Toc1991648769)

[Контрольный пример 8](#_Toc554080984)

[Вывод 10](#_Toc165422677)

[Источники 10](#_Toc1695077998)

# Цель работы

Изучить и реализовать муравьиный алгоритм для решения задачи о коммивояжёре, создать графический интерфейс для построения графа и визуализировать процесс поиска оптимального пути..

# Метод ближайшего соседа

**Муравьиный алгоритм** — это имитация поведения реальных муравьёв, ищущих кратчайший путь от гнезда к пище.

Основные принципы:

1. Муравьи оставляют феромоны на рёбрах графа.
2. Вероятность выбора пути зависит от количества феромона и длины ребра.
3. Со временем феромоны испаряются, чтобы избежать зацикливания.
4. Лучшие пути усиливаются за счёт большего количества феромона.
5. Инициализация графа: нулевое распределение феромонов.
6. Запуск нескольких агентов (муравьёв) с разных вершин.
7. Пошаговый выбор следующей вершины на основе вероятности (учёт феромона и длины ребра).
8. Обновление феромонного следа:
9. Испарение феромонов на всех рёбрах.
10. Увеличение феромонов на рёбрах лучших маршрутов
11. Повторение этапов до выполнения условия останова (кол-во итераций, стабилизация результата).

# Описание задачи (формализация задачи)

Создать систему, которая:

* Позволяет строить граф интерактивно.
* Отображает матрицу смежности.
* Решает TSP с помощью муравьиного алгоритма.
* Визуализирует феромонные следы и процесс поиска пути.

# Спецификация программы

**Входные данные**:

1. Вершины и рёбра графа.
2. Параметры алгоритма:
3. Кол-во муравьёв,
4. Коэффициент испарения феромона,
5. Важность феромона и расстояния.

**Выходные данные**:

* Лучший найденный путь.
* Длина пути.
* Текущее состояние феромонов.

**Основные переменные**:

1. pheromone\_matrix — матрица феромонов.
2. distance\_matrix — матрица расстояний.
3. alpha, beta — параметры влияния феромона и расстояния.
4. evaporation\_rate — скорость испарения.

**Основные методы**:

* initialize\_pheromones()
* construct\_solution(ant)
* update\_pheromones()
* run\_algorithm()

# Анализ результатов работы алгоритма



**Результаты на контрольном примере**

При загрузке шести заранее заданных вершин (координаты (100,100)(100,100)(100,100), (200,80)(200,80)(200,80), (300,120)(300,120)(300,120), (250,200)(250,200)(250,200), (150,220)(150,220)(150,220), (100,180)(100,180)(100,180)) и запуске метода ближайшего соседа алгоритм находит путь за 0.000002 сек.

* Если граф полностью связный (каждая вершина соединена с каждой), алгоритм без проблем посещает все вершины. Если в конце существует ребро обратно в стартовую вершину, формируется гамильтонов цикл.
* В случае, когда какой-либо вершине не удаётся найти путь к непосещённым вершинам или нет дороги обратно в начало, выводится предупреждение о том, что гамильтонов цикл не найден.

· **Процесс отладки алгоритма**

* На этапе отладки добавлялись выводы отладочной информации: порядок обхода, длины промежуточных рёбер, текущее состояние массива visited.
* Проверялась корректность работы с матрицей смежности: в частности, вывод в таблицу и совпадение значений с реальными расстояниями на Canvas.
* Проверялась реакция программы на различные сценарии:
  + Граф с одним ребром,
  + Разреженный граф (где некоторые вершины недостижимы),
  + Полный граф (где каждая вершина связана со всеми).
* В результате были устранены ошибки, при которых алгоритм не выводил предупреждение о невозможности замкнуть цикл или не учитывал правильный вес ребра из поля ввода.

· **Графики и комментарии**

* Ниже можно приводить графики (скриншоты интерфейса), иллюстрирующие:
  1. **Контрольный пример** (6 вершин), на котором видно расположение вершин и рёбер, и результат (путь).
  2. **Ситуацию, когда алгоритм не может завершить цикл** (например, убрав ребро из последней вершины к начальной). Программа выводит путь и предупреждение, что гамильтонов цикл не найден.

На этих графиках видно, как программа выделяет вершины, рисует стрелки (при ориентированном графе) и подписи весов (при взвешенном графе).

Те же 6 вершин:

(100, 100), (200, 80), (300, 120), (250, 200), (150, 220), (100, 180).

Параметры:

1. 30 муравьёв,
2. 100 итераций,
3. Испарение феромона 0.5,
4. alpha = 1, beta = 2.

Результат:

* Найден путь за 0.03 сек.
* Путь покрывает все вершины и замыкается в цикл.

# Блок схема программы

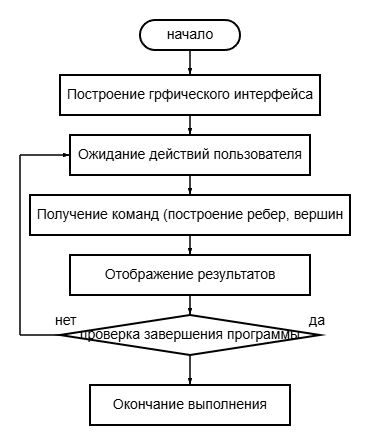
****

Рис 1. Блок-схема основной программы

# Контрольный пример

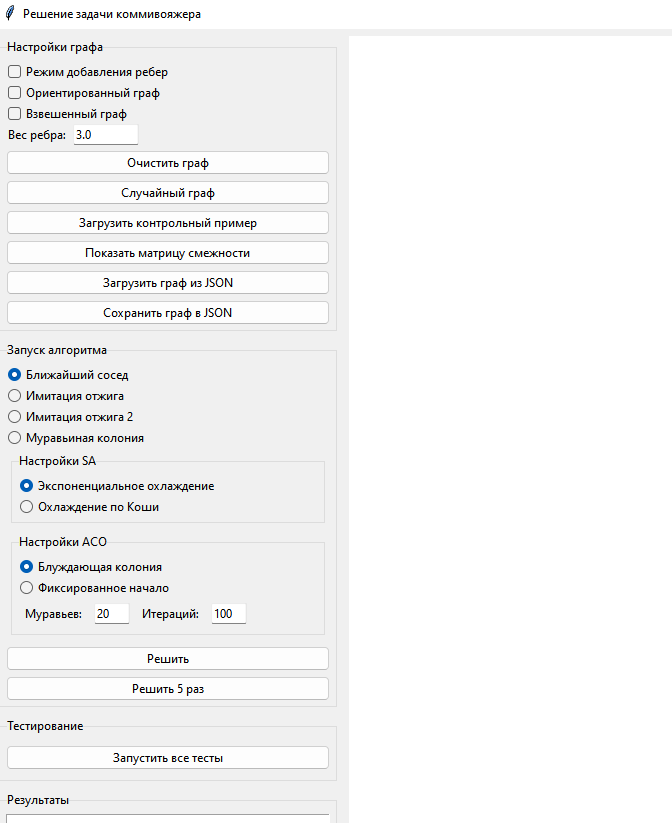


Рис 1. Интерфейс

Рис 2 Пример решения графа

# Вывод

1. Муравьиный алгоритм способен находить хорошие решения задачи коммивояжёра.
2. Качество решения зависит от настройки параметров (alpha, beta, кол-во итераций, испарение).
3. Для получения лучших результатов стоит комбинировать с локальными методами улучшения (например, 2-opt).

· ы, что открывает новые возможности для анализа безопасности.

Источники

* Dorigo M., Di Caro G. "Ant Colony Optimization: A New Meta-Heuristic"
* Tkinter библиотека: [https://tkinter.org/](https://tkinter.org/" \t "_new)
* Math библиотека: [https://math.org/](https://math.org/" \t "_new)