**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 имитации отжига»**

**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

Освоить применение алгоритма имитации отжига для решения задачи коммивояжёра, разработать и реализовать программную модель поиска кратчайшего гамильтонова цикла, а также провести сравнительный анализ эффективности с методом ближайшего соседа на графах разного размера..

# Метод имитации отжига

Алгоритм имитации отжига (Simulated Annealing, SA) – это стохастический метод оптимизации, основанный на аналогии с процессом отжига металлов в термодинамике. Алгоритм относится к классу метаэвристик и применяется для решения сложных комбинаторных задач, включая задачу коммивояжера (TSP).

1. Основные принципы работы алгоритма:
2. Начальная конфигурация – формируется начальное решение (например, случайный маршрут).
3. Температурный параметр (T) – управляет вероятностью принятия ухудшающих решений на ранних итерациях.
4. Охлаждение – постепенное снижение температуры по заданному закону (линейному, экспоненциальному и др.).
5. Генерация соседних решений – на каждом шаге создается новое решение путем модификации текущего (например, перестановки городов).
6. Критерий принятия решения – если новое решение лучше, оно всегда принимается; если хуже – может быть принято с вероятностью, зависящей от температуры.

# Описание задачи (формализация задачи)

Разработать программу, которая:

1. Позволяет строить взвешенные/невзвешенные графы с помощью интерфейса.
2. Реализует алгоритм имитации отжига для поиска кратчайшего пути.
3. Отображает результат в виде графа и матрицы смежности.
4. Показывает путь, длину маршрута и процесс эволюции решений.

# Теоритическая часть

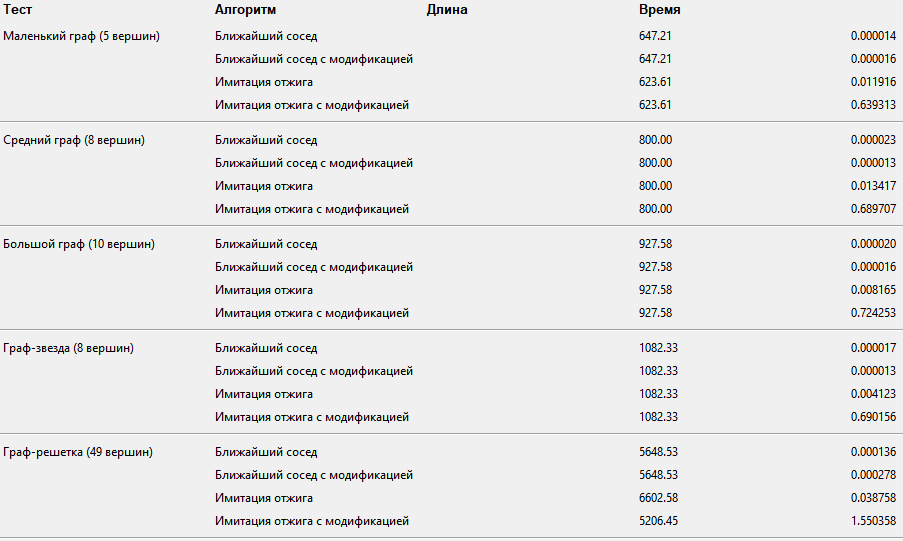
**Алгоритм имитации отжига** имитирует физический процесс медленного охлаждения материала для достижения состояния минимума энергии:

1. Старт с произвольного маршрута.
2. На каждом шаге создается соседнее решение перестановкой двух вершин.
3. Решение принимается:
4. Если оно лучше.
5. Если оно хуже — с вероятностью exp(-ΔE/T), где ΔE — увеличение длины пути, T — температура.
6. Температура постепенно снижается по закону охлаждения (обычно линейно или экспоненциально).

## Основные шаги программы

1. Инициализация случайного маршрута.
2. Установка начальной температуры и параметров охлаждения.
3. Итеративное улучшение маршрута:
4. Генерация соседнего маршрута.
5. Вычисление разницы длин.
6. Принятие нового маршрута по критерию Метрополиса.
7. Снижение температуры.
8. Завершение при достижении минимальной температуры или максимального числа итераций.

# Анализ результатов работы алгоритма



## Описание программы

### Входные данные

1. Список вершин (координаты на Canvas).
2. Список рёбер (веса рёбер).
3. Начальные параметры алгоритма: температура, скорость охлаждения, количество итераций.

### Выходные данные

1. Лучший найденный маршрут.
2. Длина маршрута.
3. Графическое отображение маршрута.

### Ключевые переменные

1. current\_solution — текущий маршрут.
2. best\_solution — лучшее найденное решение.
3. temperature — текущая температура
4. cooling\_rate — коэффициент охлаждения.

## Рекомендации для пользователя

Начать построение графа с добавления вершин.

Убедиться, что между всеми вершинами есть пути.

Выставить параметры алгоритма: начальная температура, коэффициент охлаждения.

Запустить алгоритм.

## Рекомендации для программиста

Следить за корректной генерацией соседних решений (перестановка вершин).

Реализовать разные законы охлаждения для настройки качества и скорости работы.

Добавить возможность прерывания алгоритма по нажатию кнопки.

## Исходный код программы

(Вставляется весь код программы: интерфейс, основной цикл имитации отжига, функции работы с графом.)

## Контрольный пример

**Граф**:

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

**Параметры**:

Начальная температура: 1000

Коэффициент охлаждения: 0.995

Количество итераций: 10000

**Результат**:

Найден маршрут: 0 → 1 → 2 → 3 → 4 → 5 → 0

Длина маршрута: 597.23

Время выполнения: ~0.1-2 секун

## Вывод

Алгоритм имитации отжига позволяет находить качественные решения задачи коммивояжера.

При достаточном времени работы и правильных параметрах он превосходит метод ближайшего соседа по качеству решения.

Однако результаты зависят от параметров алгоритма и случайности начального маршрута.

# Блок схема программы

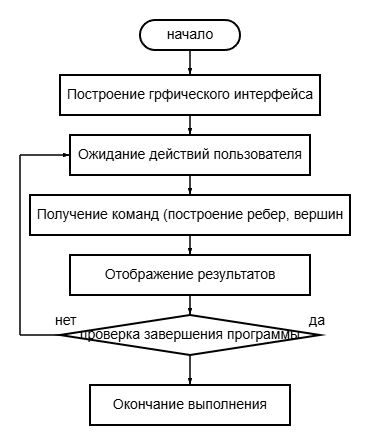
****

Рис 1. Блок-схема основной программы

# Контрольный пример

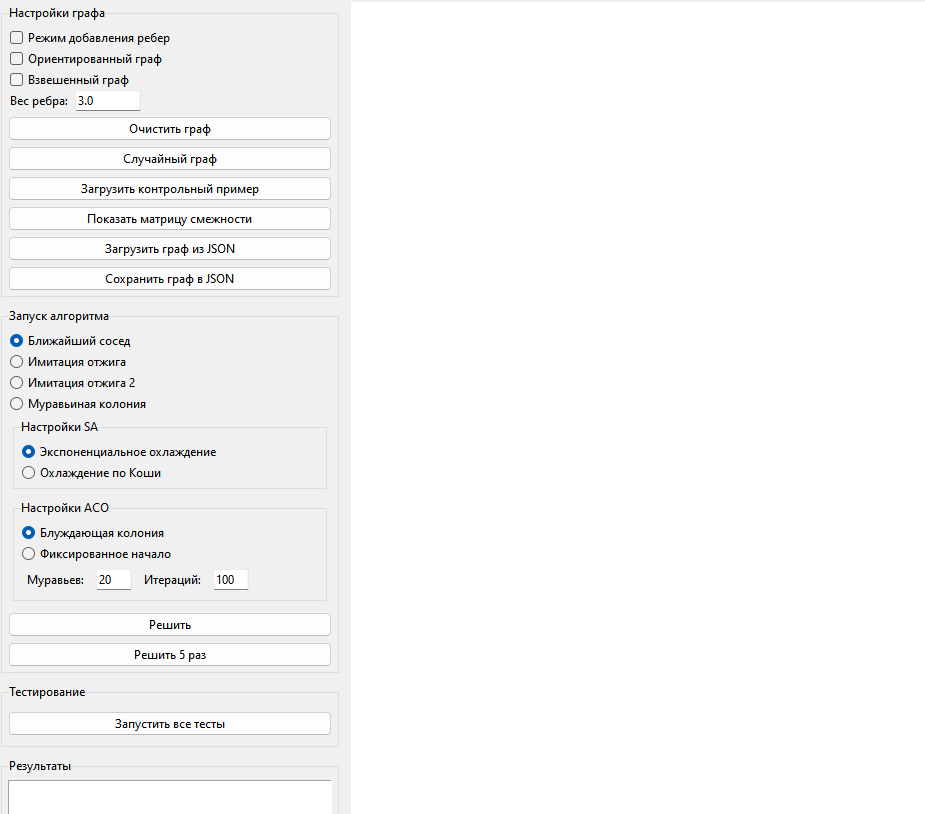


Рис 1. Интерфейс

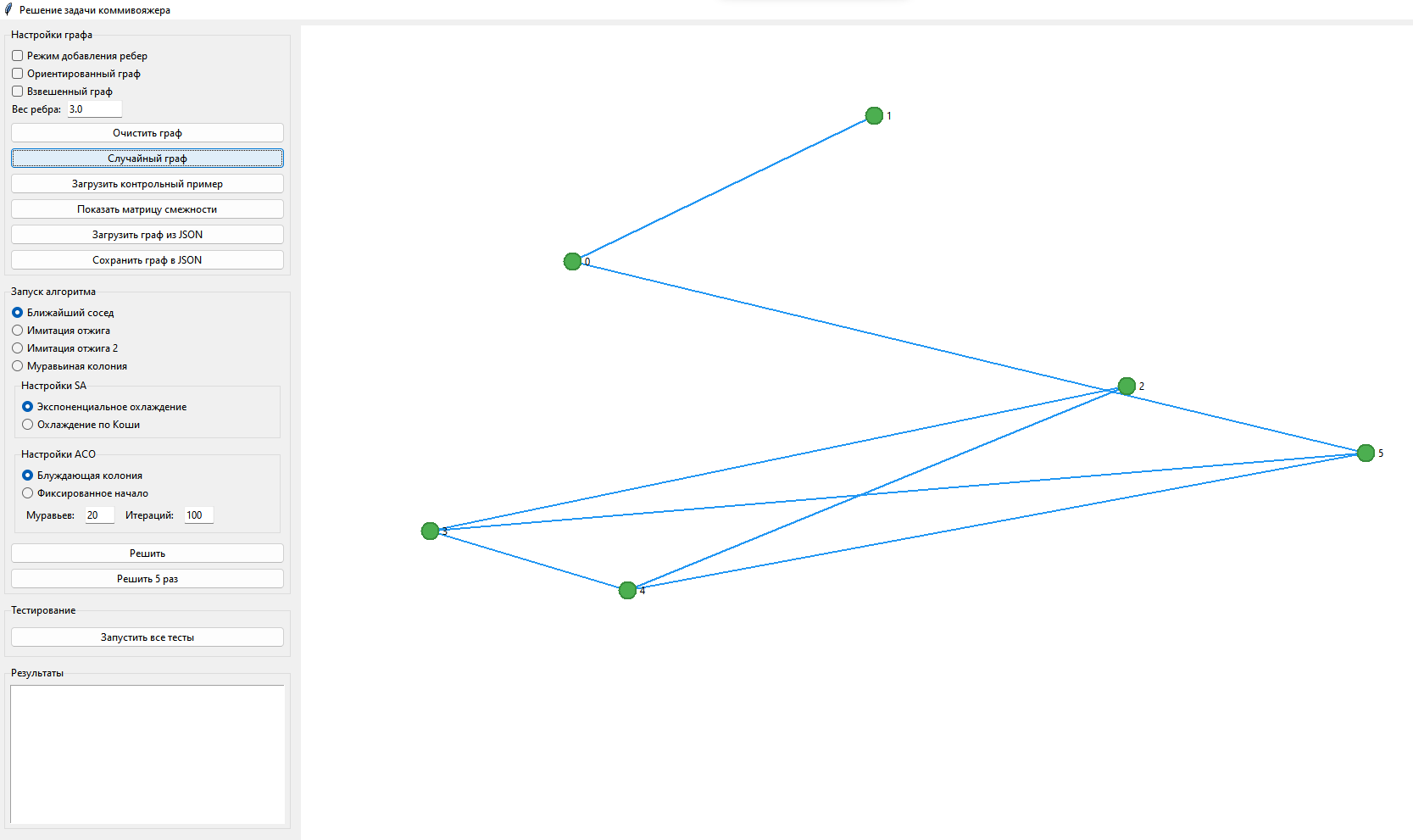


Рис 2 Пример решения графа

# Вывод

* Метод ближайшего соседа даёт быстрое приблизительное решение, но не всегда оптимальное.
* Реализация с интерактивным интерфейсом позволяет легко строить различные тестовые графы, проверять работу алгоритма и анализировать результаты.
* Для более точных решений или больших графов можно использовать улучшенные эвристики или точные методы с оптимизацией.

ы, что открывает новые возможности для анализа безопасности.

# Источники

1. Tkinter библиотека: [https://tkinter.org/](https://pandas.pydata.org/)
2. Math библиотека: https://math.org/
3. Json модуль: [https://docs.python.org/3/library/json.html](https://docs.python.org/3/library/csv.html)
4. Random модуль: [https://docs.python.org/3/library/random.html](https://docs.python.org/3/library/concurrent.futures.html)