САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики-процессов управления

Программа бакалавриата
“Большие данные и распределенная цифровая платформа”

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

на тему «Решение задачи о коммивояжере с помощью алгоритма имитации отжига»

Модификация – «Сверхбыстрый отжиг»

Студент гр. 23Б15-пу
Сериков К.Г.

Преподаватель
Дик А.Г.

Санкт-Петербург
2025 г.

Оглавление

Цель работы

Целью данной лабораторной работы является изучение и реализация алгоритма имитации отжига для решения задачи коммивояжёра (TSP). В рамках данной работы была разработана программа, которая находит кратчайший гамильтонов цикл при помощи алгоритма имитации отжига и предоставляет возможность переключения между обычной и модифицированной версиями алгоритма.

Описание задачи (формализация задачи)

В данной лабораторной работе необходимо исследовать эффективность алгоритма имитации отжига для решения задачи коммивояжёра (TSP).

Формализация задачи включает следующие компоненты:

1. **Изучение особенностей алгоритма имитации отжига:** Исследовать принципы работы алгоритм имитации отжига, включая его применение для решения задачи коммивояжера. Рассмотреть эффективность алгоритма.
2. **Разработка алгоритма для нахождения кратчайшего гамильтонова цикла:** Создать программу, реализующую алгоритм имитации отжига для поиска решения задачи коммивояжера. Реализовать визуализацию исходного и итогового графов.
3. **Тестирование программы на взвешенном ориентированном графе:** Провести тестирование программы на контрольных примерах. Оценить точность и эффективность метода имитации отжига для решения задачи коммивояжёра (TSP).
4. **Анализ эффективности алгоритма**: Провести сравнительный анализ работы алгоритма имитации отжига с модификацией и без неё, а также с алгоритмом ближайшего соседа. Исследовать влияние структуры графа на эффективность. Оценить влияние модификации на скорость и точность нахождения кратчайшего гамильтонова цикла.

Теоретическая часть

Задача коммивояжера

Задача коммивояжёра (Traveling Salesman Problem) является одной из классических задач комбинаторной оптимизации. Она формулируется следующим образом: имеется множество городов и известны расстояния между каждой парой городов. Требуется найти кратчайший маршрут, который проходит через каждый город ровно один раз и возвращается в начальную точку.

Алгоритм имитации отжига

Алгоритм имитации отжига — эвристический алгоритм глобальной оптимизации. Алгоритм вдохновлён процессом отжига в металлургии — техники, заключающейся в нагревании и контролируемом охлаждении металла, чтобы увеличить его кристаллизованность и уменьшить дефекты. Этот метод прост в реализации, однако далеко не всегда находит оптимальный маршрут.

Сверхбыстрый отжиг

Сверхбыстрый отжиг — это модификация классического алгоритма имитации отжига, предложенная для ускорения сходимости и повышения эффективности поиска на больших пространствах решений. Основное отличие заключается в стратегии охлаждения температуры и распределении вероятностей генерации новых решений.

В сверхбыстром отжиге применяется следующая функция охлаждения:

$$T(k) = T\_0 / (1 + k)$$

Принцип работы

1. Формируется начальный маршрут
2. Задаётся начальная температура.
3. Генерируется соседнее решение
4. Рассчитывается разница в длине маршрута
5. Температура постепенно понижается
6. Цикл повторяется до достижение максимального числа итераций

Основные шаги программы

1. **Запуск программы:** Инициализируется графический интерфейс с элементами управления.
2. **Настройка параметров:** Пользователь в интерфейсе может задать граф вручную или загрузить JSON файл, также пользователь выбирает между стандартным и модифицированным режимами алгоритма.
3. **Запуск алгоритма:** При нажатии на кнопку "Рассчитать" программа запускает выполнение алгоритма имитации отжига на заданном графе.
4. **Начало работы алгоритма:** Алгоритм случайным образом генерирует начальный путь (перестановку всех вершин графа), рассчитывает его общую длину, устанавливает начальную температуру и другие параметры отжига.
5. **Построение маршрута:** Итеративно алгоритм вносит изменения в маршрут, и решает — принять новое решение или нет — на основе температуры. Температура постепенно понижается, уменьшая вероятность принятия худших решений.
6. **Завершение цикла:** Алгоритм заканчивает выполнение, когда достигается максимальное число итераций. Лучший найденный путь сохраняется как результат.
7. **Отображение результатов:** Программа отображает кратчайший гамильтонов цикл в виде графа и длину это пути и вершины маршрута.

Блок схема программы

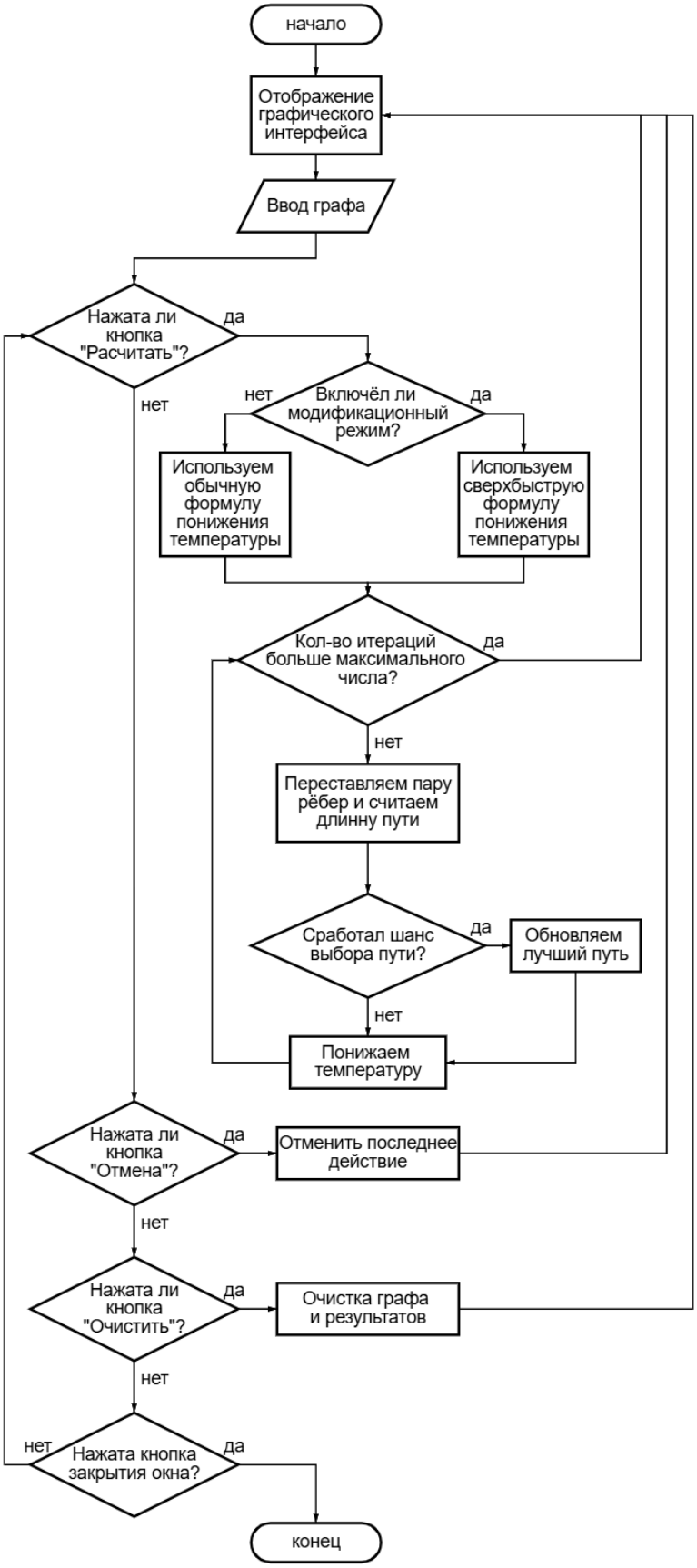

Рис 1. Блок-схема основной программы

Описание программы

Программная реализация написана на языке Python 3.13.0 с использованием следующих библиотек: tkinter [1], numpy [2], time [3], math [4], json [5] и random [6]. Программа организована в виде графического интерфейса для решения задачи коммивояжёра (TSP) с помощью алгоритма имитации отжига. В программе реализован только один класс – TSPApp, который отвечает за работу всей программы. В процессе разработки программы использовался main.py, включающий 14 функций, каждая из которых имеет чётко определённое назначение:

Таблица 1. main.py

Функция	Описание	Возвращаемое значение
__init__	Инициализация интерфейса.	None
add_node_or_edge	Добавляет вершину или ребро в зависимости от места клика	None
add_node	Добавляет новую вершину в заданной позиции	None
add_edge	Добавляет ребро между двумя вершинами с вычислением евклидова расстояния	None
find_clicked_node	Находит вершину по координатам клика	int или None
redraw_graph	Перерисовывает граф и обновляет таблицу рёбер	None
get_distance_matrix	Создает матрицу расстояний между всеми вершинами графа	numpy.ndarray
calculate_path_length	Вычисляет общую длину гамильтонова цикла	float
solve_tsp	Реализует алгоритм ближайшего соседа	None

visualize_solution	Визуализирует найденный маршрут на отдельном холсте	None
load_graph	Загружает граф из JSON-файла	None
get_distance	Возвращает расстояние между двумя вершинами	float
undo_action	Отменяет последнее действие	None
clear_graph	Полностью очищает граф, историю изменений и результаты	None

Рекомендации пользователя

Программа позволяет запустить алгоритма имитации отжига для решения задачи коммивояжёра (TSP) с использованием графического интерфейса.

1. **Запуск программы**: Откройте программу с помощью Python 3.13.0, чтобы инициализировать интерфейс. Интерфейс позволяет нарисовать граф и выбирать режим: стандартный или модифицированный.
2. **Настройка параметров**: В графическом интерфейсе реализованы следующий функции:
   - Добавление узлов: Для создания нового узла кликните левой кнопкой мыши на область графа.
   - Добавление ребер: Для создания ребра между узлами кликните сначала на один узел, затем на второй. Длина ребра будет автоматически подсчитана.
   - Выбор модификации: Для использования модификации «Сверхбыстрый отжиг» установите галочку в флажок «Использовать модификацию».
3. **Запуск алгоритма**: Нажмите кнопку «Рассчитать» для выполнения алгоритма. При этом будет вычислен и отображён кратчайший гамильтонов цикл. Программа отобразит последовательность вершин кратчайшего гамильтонова цикла и построенный маршрут с выделенными ребрами.

4. **Дополнительные элементы интерфейса**: Для отмены последнего действия нажмите кнопку «Отмена». Для сброса графа нажмите кнопку «Очистить».

Рекомендации программиста

Актуальность версии Python: Используйте обновленную версию Python, tkinter [1], numpy [2], time [3], math [4], json [5] и random [6]. Уделяйте внимание четкому именованию переменных и функций. Тщательно тестируйте на различных графах, чтобы удостовериться в корректной работе алгоритма и визуализации. Убедитесь, что интерфейс программы в Tkinter легко понимается пользователями. Разделите входные данные, управление алгоритмом и отображение результатов по разным секциям.

Исходный код программы

https://github.com/romplle/spbu-algorithms-and-data-structures/

Контрольный пример

1. Запуск программы и ввод параметров

   Для запуска программы откройте main.py. Программа откроет графический интерфейс для настройки и запуска алгоритма имитации отжига, предназначенного для решения задачи коммивояжёра (TSP) (Рис. 2).

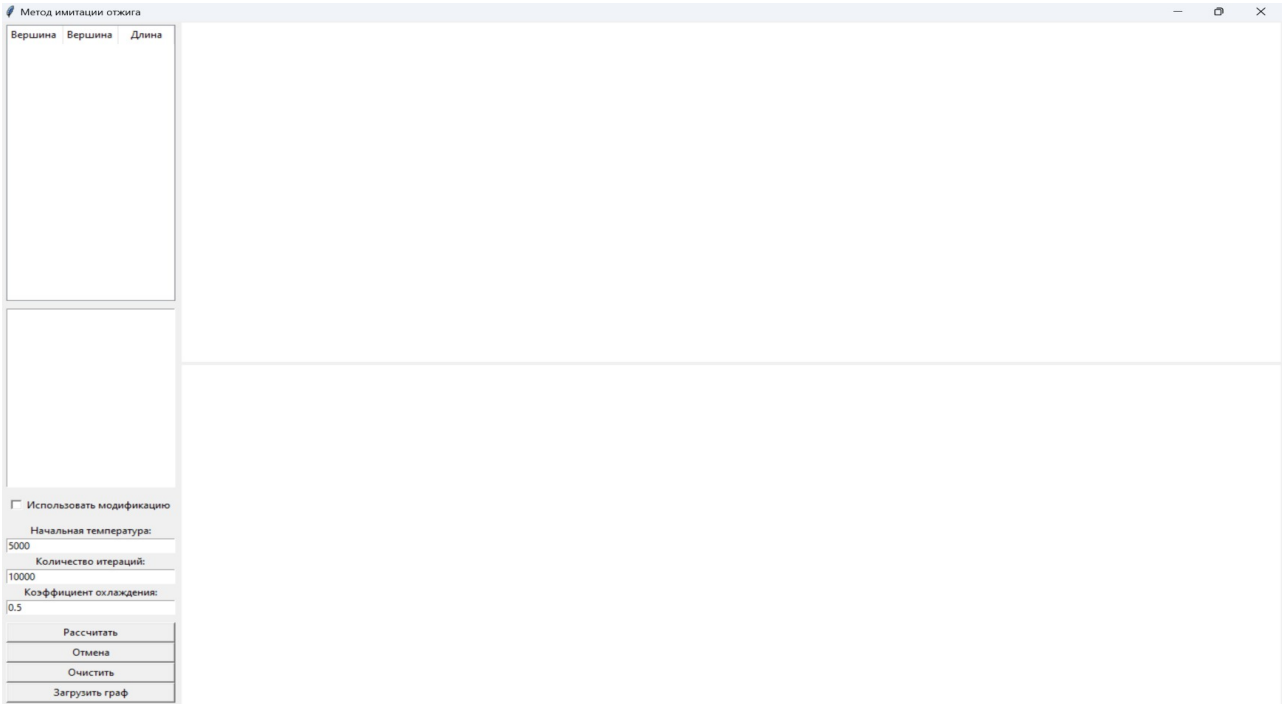

Рис. 2. Начальное окно программы

2. Ввод исходный данных

   После запуска программы можно добавить вершины и соединить их ребрами вручную, а можно нажать кнопку «Загрузить граф» и выбрать JSON-файл для импортирования готового графа. Вы также можете выбрать режим работы (стандартный или модифицированный) (Рис. 3).

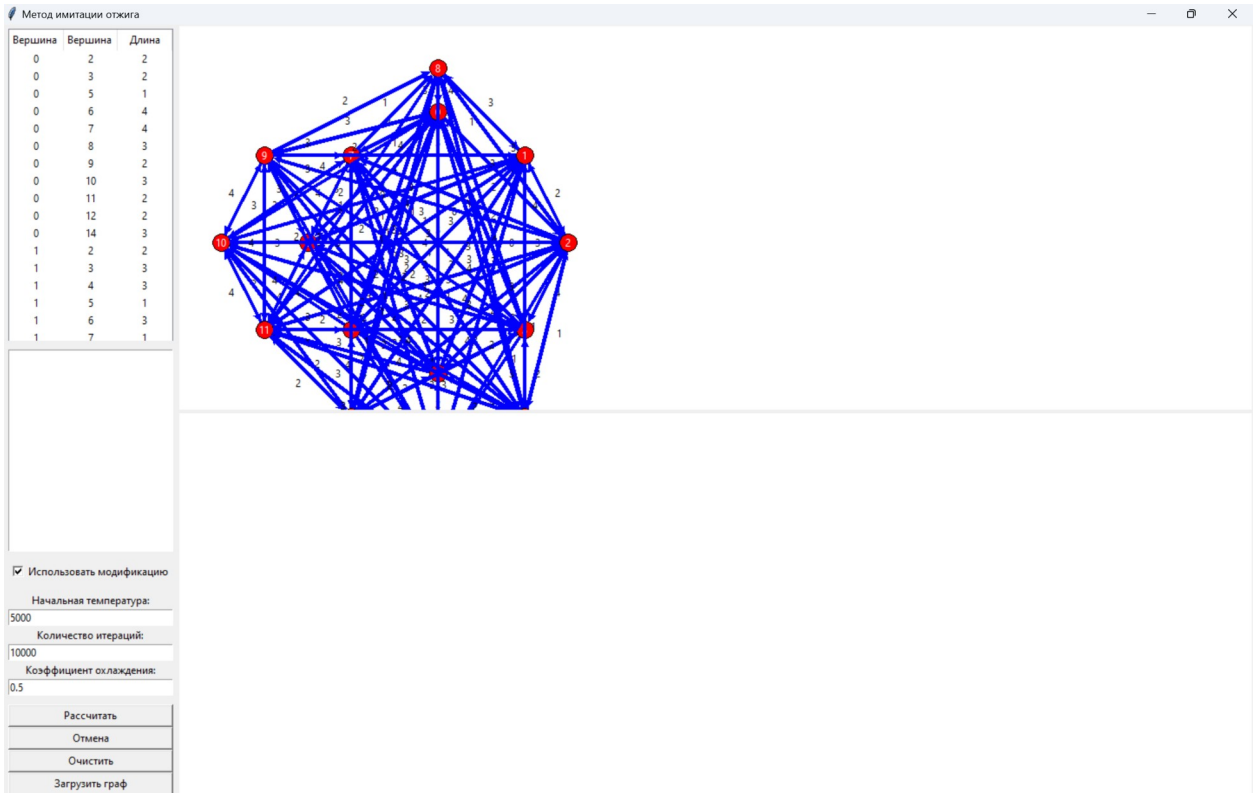

Рис. 3. Пример исходных данных программы

3. Запуск алгоритма

После задания графа нажмите кнопку «Рассчитать», чтобы запустить алгоритм. Программа вычислит кратчайший гамильтонов цикл методом имитации отжига, обновляя интерфейс для отображения результатов (Рис. 4).

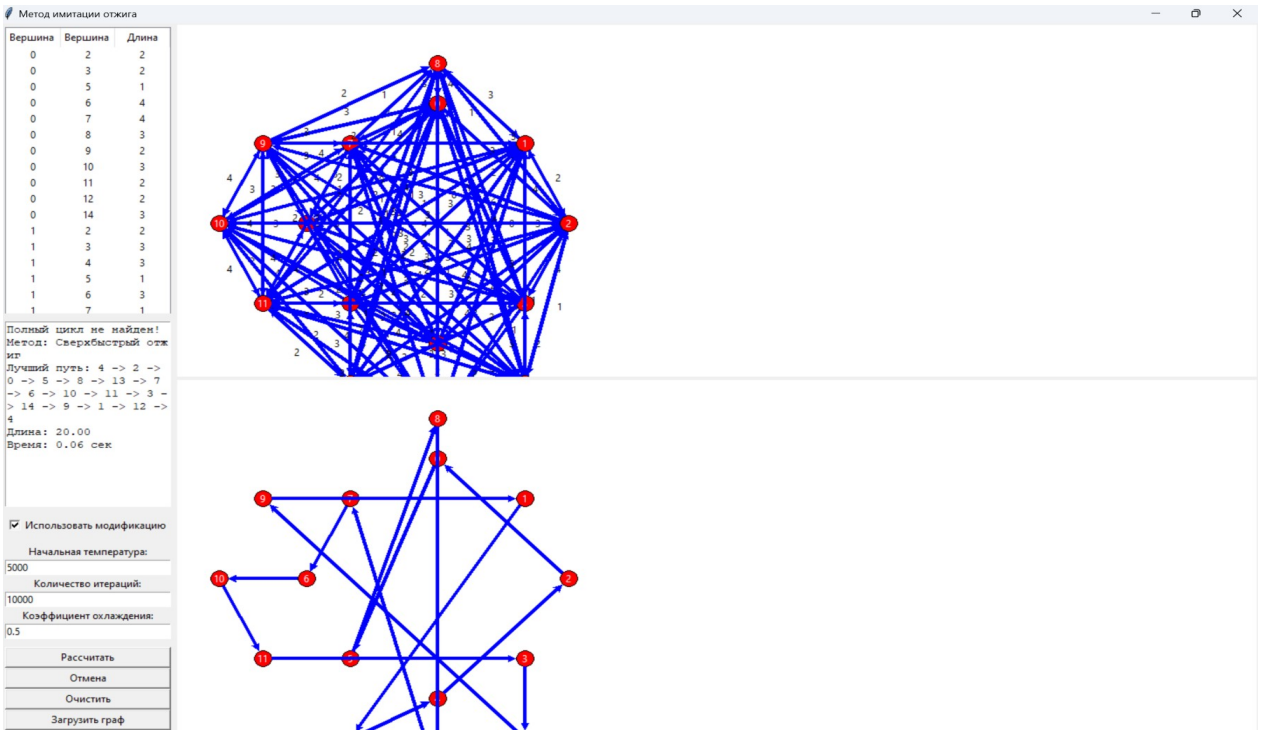

Рис. 4. Пример результатов программы

Анализ

В качестве модификации классического алгоритма имитации отжига был применён метод сверхбыстрого отжига. Для сравнения также был протестирован алгоритм ближайшего соседа. Основной целью являлось сравнение времени выполнения и качества найденного маршрута (длины пути) в зависимости от количества вершин и использования модификации.

Начальные параметры: начальная температура — 5000, количество итераций — 10000, коэффициент охлаждения — 0.5.

Таблица 2: тестирование алгоритма имитации отжига без модификации

Кол-во вершин	Модификация	Время	Длина пути
6	Без модификации	0.07 сек	12
6	Без модификации	0.07 сек	13
6	Без модификации	0.07 сек	16
15	Без модификации	0.09 сек	28
15	Без модификации	0.09 сек	25
15	Без модификации	0.09 сек	24
25	Без модификации	0.12 сек	29
25	Без модификации	0.13 сек	32
25	Без модификации	0.12 сек	31

Таблица 3: тестирование алгоритма имитации отжига с модификацией

Кол-во вершин	Модификация	Время	Длина пути
6	С модификацией	0.07 сек	12
6	С модификацией	0.07 сек	12
6	С модификацией	0.07 сек	15
15	С модификацией	0.09 сек	21
15	С модификацией	0.09 сек	24
15	С модификацией	0.09 сек	22
25	С модификацией	0.12 сек	27
25	С модификацией	0.12 сек	27
25	С модификацией	0.11 сек	26

Таблица 4: тестирование алгоритма ближайшего соседа

Кол-во вершин	Модификация	Время	Длина пути
6	Без модификации	0.0001 сек	12
6	С модификацией	0.0002 сек	12
15	Без модификации	0.0022 сек	18
15	С модификацией	0.0258 сек	17
25	Без модификации	0.0090 сек	31
25	С модификацией	0.1414 сек	25

Общие наблюдения

Влияние модификации (сверхбыстрого отжига):
В большинстве случаев модифицированный алгоритм (с использованием метода сверхбыстрого отжига) показал лучшие результаты по длине маршрута по сравнению с классическим вариантом. Особенно заметна разница на графах из 15 и 25 вершин.

Время выполнения алгоритма имитации отжига как с модификацией, так и без неё оказалось примерно одинаковым для одного и того же числа итераций и вершин.

Алгоритм ближайшего соседа демонстрирует значительно более высокую скорость, особенно на малых графах (например, 6 и 15 вершин). Однако по качеству маршрутов он уступает обоим вариантам имитации отжига, особенно на графах из 25 вершин.

Модифицированный имитационный отжиг с применением сверхбыстрого охлаждения представляет собой наиболее сбалансированный подход, обеспечивая хорошее качество решений при разумном времени выполнения. Он превосходит классический отжиг по эффективности и работает более стабильно, чем жадный метод ближайшего соседа, особенно при увеличении размера задачи.

Вывод

В рамках работы был реализован алгоритм имитации отжига для решения задачи коммивояжёра, модифицированный путём сверхбыстрого отжига. Также был разработан удобный графический интерфейс. Преимуществами алгоритма являются простота реализации и высокая вычислительная эффективность, что делает его подходящим для быстрого приближённого решения задачи.

Источники

1. tkinter — Python interface to Tcl/Tk // URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 26.03.2025).
2. NumPy documentation // URL: https://numpy.org/doc/stable/index.html (дата обращения: 26.03.2025).
3. time — Time access and conversions // URL: https://docs.python.org/3/library/time.html (дата обращения: 27.03.2025).
4. math — Mathematical functions // URL: https://docs.python.org/3/library/math.html (дата обращения: 27.03.2025).
5. json — JSON encoder and decoder // URL: https://docs.python.org/3/library/json.html (дата обращения: 27.03.2025).
6. random — Generate pseudo-random numbers // URL: https://docs.python.org/3/library/random.html (дата обращения: 10.04.2025).