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 метода имитации отжига»

Студент гр. 23Б15-пу
Антонян А. А.

Преподаватель
Дик А.Г.

Санкт-Петербург
2025 г.

Оглавление

Цель работы

Целью лабораторной работы является решение задачи о коммивояжере с помощью алгоритма, использующего метод имитации отжига с модификацией Коши и без неё.

Теоретическая часть

Метод имитации отжига — один из эвристических подходов решения задачи коммивояжёра, основанный на аналогии с физическим процессом отжига металлов. В отличие от простых жадных алгоритмов (таких как метод ближайшего соседа), метод имитации отжига позволяет выходить из локальных минимумов за счёт случайных изменений текущего решения и возможности принятия решений, ухудшающих текущий результат, с некоторой вероятностью. Вероятность принятия такого решения уменьшается со временем по мере снижения температуры системы.
Модификация алгоритма — отжиг Коши — отличается от классической реализации способом понижения температуры. В обычном отжиге температура уменьшается экспоненциально (T = T * alpha, где alpha < 1), а при отжиге Коши используется более медленное уменьшение температуры по закону Коши (T = T / (1 + T)), что позволяет алгоритму дольше исследовать пространство решений и повышает шансы на нахождение оптимального или близкого к нему решения.
Метод имитации отжига и его модификация Коши обеспечивают лучшее качество решений по сравнению с простыми жадными алгоритмами, однако требуют больше вычислительных ресурсов и времени на выполнение.

Основные принципы:

1. Веса рёбер графа направленные и фиксированные.
2. На каждом шаге алгоритм случайно выбирает новую конфигурацию (путь).

3. Новая конфигурация принимается всегда, если она лучше текущей, или с некоторой вероятностью, зависящей от текущей температуры, если она хуже.
4. Температура постепенно понижается, снижая вероятность принятия невыгодных решений.

Описание задачи

Задача коммивояжера заключается в поиске гамильтонова цикла минимальной длины в графе. Веса ребер могут меняться в зависимости от направления.

Основные шаги программы

1. Выбор начального состояния: случайно выбирается начальный путь, охватывающий все узлы графа.
2. Расчёт длины текущего маршрута: подсчитывается суммарный вес выбранного пути. Если между какими-то двумя последовательными узлами ребра нет, маршрут считается недопустимым и его длина — бесконечной.
3. Итеративное улучшение маршрута (процесс отжига):
    a. Устанавливается начальная температура.
    b. Повторяются следующие действия, пока температура не достигнет заданного минимума:
        i. На каждой итерации:
            1. Создаётся новое состояние путём случайного изменения текущего маршрута (перестановкой или инверсией части пути)
            2. Рассчитывается длина нового маршрута.
            3. Если новое состояние лучше текущего, оно принимается.
            4. Если новое состояние хуже текущего, оно принимается с вероятностью, зависящей от текущей

температуры и разницы длин маршрутов: P = exp(-delta/T), где delta — разница длин нового и текущего маршрутов, а T — текущая температура.

5. Понижение температуры:
    a. Обычный отжиг: T = T * alpha, где alpha < 1
    b. Отжиг Коши: T = T / (1 + T).

4. Завершение алгоритма:
    a. Итерации прекращаются, когда температура достигает заданного минимального порога.
    b. Если маршрут корректный, то он принимается как решение.

Рис. 1. Блок-схема алгоритма

Описание программы

Программная реализация написана на языке Python 3.12.2 с использованием библиотек PyQt5[1], matplotlib[2] и networkx[3]. В процессе разработки программы использовался следующий модуль:

Таблица1 simulated_annealing.py

Функция	Описание	Возвращаемое значение
simulated_annealing(start, use_cauchy)	Решает задачу коммивояжёра методом имитации отжига.	tuple
nearest_neighbor(start)	Решает задачу коммивояжёра методом ближайшего соседа.	tuple
path_distance(path)	Рассчитывает длину заданного пути.	Int или float(inf)
generate_random_graph()	Создаёт случайный ориентированный взвешенный граф заданного размера.	None
add_node(x, y)	Добавляет узел в граф по координатам.	None
add_edge(node1, node2)	Добавляет направленное ребро между двумя узлами со случайным весом.	None
run_algorithm()	Выполняет выбранный алгоритм и выводит результат.	None
toggle_annealing(state)	Переключает режим выполнения алгоритма на имитацию отжига.	None
toggle_nn(state)	Переключает режим выполнения алгоритма на метод ближайшего соседа.	None
toggle_cauchy(state)	Включает или выключает модификацию отжига Коши.	None

Рекомендации пользователя

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как PyQt5[1] и matplotlib[2] и networkx[3]. Код можно запустить в среде разработки или через командную строку, используя консоль для настройки параметров и генерации данных. Запуск программы производится через файл simulated_annealing.py.

При запуске программы вам будет предложено выбрать параметры запускаемого алгоритма. Вводите ответы соответствующие поля в GUI.

Рекомендации программиста

Для поддержания актуальности и работоспособности программы используйте последние версии библиотек, особенно PyQt5[1] и matplotlib[2] и networkx[3]. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода.

Исходный код программы:

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

Контрольный пример

1. **Запуск программы**
   Для запуска программы используйте файл simulated_annealing.py. Программа загружает GUI и позволяет пользователю настроить параметры алгоритма.

2. **Выбор параметров**
   После запуска программы пользователю будет предложено выбрать параметры (Рис. 2).

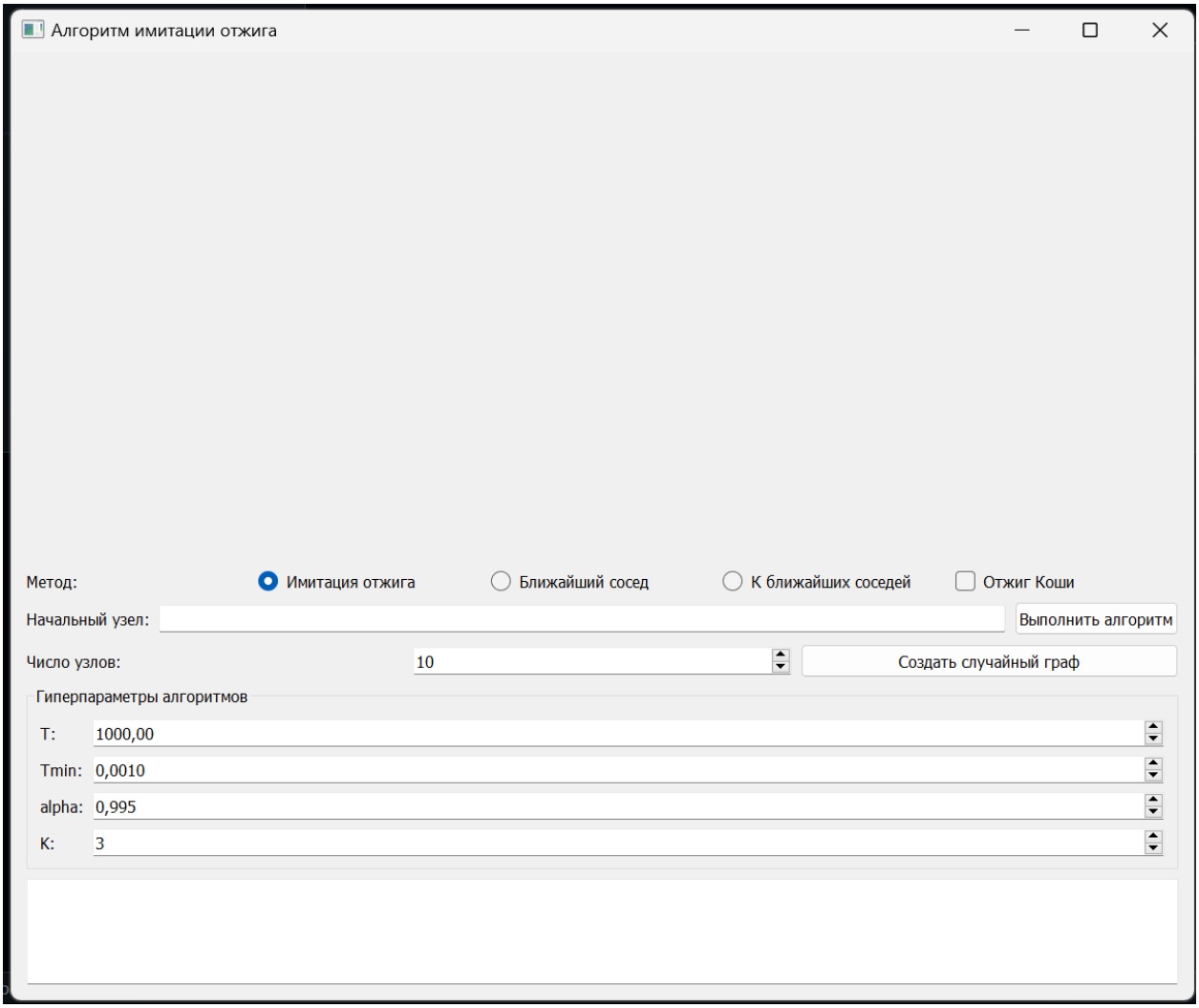

Рис 2. Пример выбора параметров

3. **Обработка данных и вывод результатов**

После выбора параметров пользователю предложено запустить алгоритм. Результат работы будет выводиться в окнах ниже (Рис. 3).

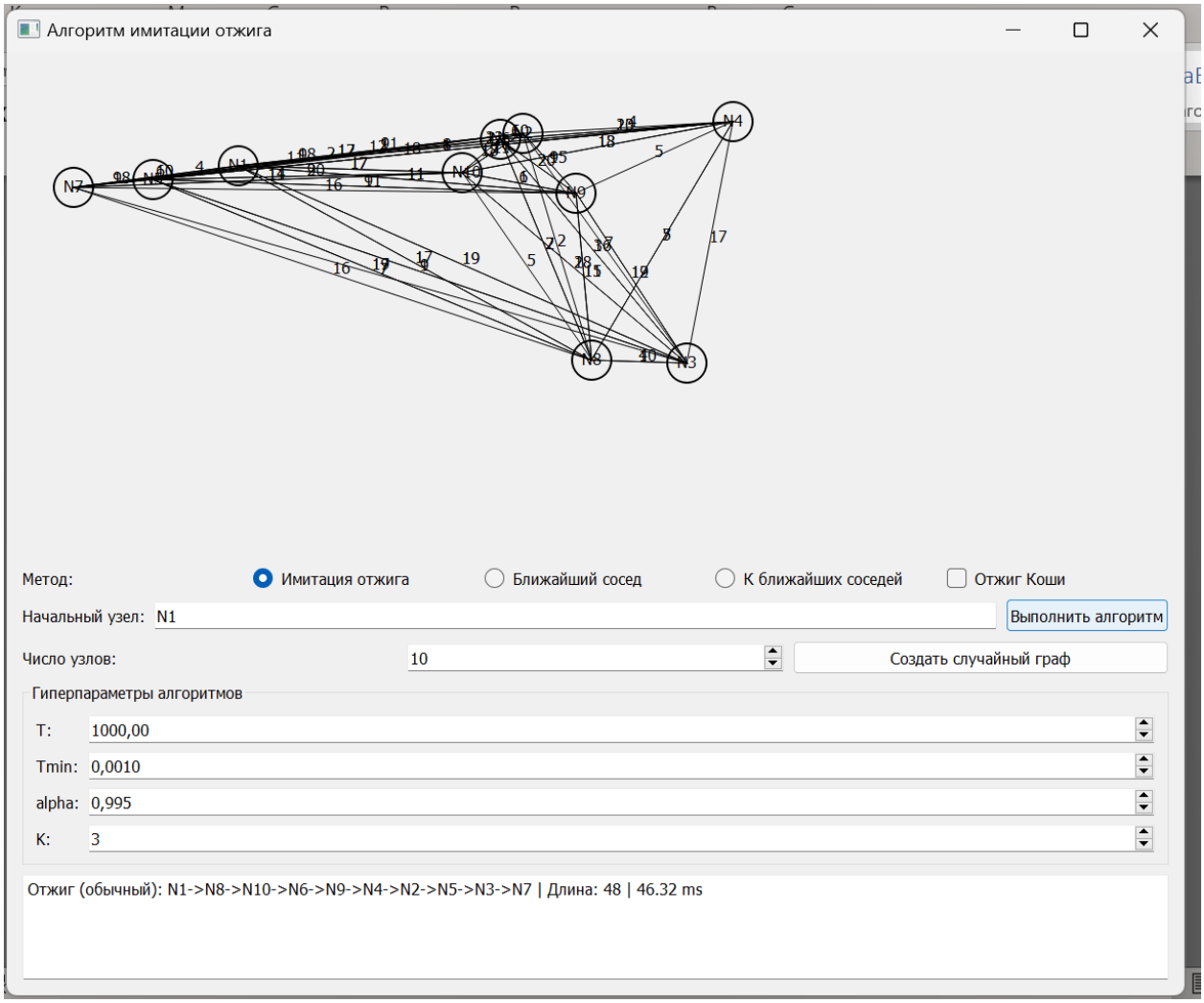

Рис 3. Результат работы алгоритма

Исследование

В рамках данной лабораторной работы был создан алгоритм решения задачи коммивояжера методом имитации отжига с модификацией Коши и без. В ходе тестирования программы было выявлено, что модифицированный алгоритм работает намного медленнее обычного, но иногда находит более оптимальные пути (за счет замедленного понижения температуры). В ходе сравнения данного алгоритма с алгоритмом метода ближайшего соседа было выявлено, что метод имитации отжига работает лучше метода ближайшего соседа и метода K ближайших соседей, но время работы значительно выше. Оптимальные параметры для имитации алгоритма отжига: T = 1000, Tmin = 0,001, alpha = 0,995.

Таблица 2. Сравнение алгоритмов

Количество вершин	Метод имитации отжига (время выполнения)	Метод имитации отжига (длина пути)	Метод имитации отжига Коши (время выполнения)	Метод имитации отжига Коши (длина пути)	Метод ближайшего соседа (время выполнения)	Метод ближайшего соседа (длина пути)	Метод K ближайших соседей (время выполнения)	Метод K ближайших соседей (длина пути)
10	48.02 мс	49,3	1503.04 мс	38,1	0 мс	66	1.0 мс	41
20	78.55 мс	116,7	2169.32 мс	102,4	1.0 мс	120	1.0 мс	110
30	85.2 мс	187,2	2896.11 мс	164,9	1.0 мс	205	1.01 мс	195

Вывод

В рамках данной работы был разработан алгоритм для нахождения кратчайшего пути в задаче коммивояжера. Реализованный алгоритм обеспечивает нахождение различных путей – оптимальных и нет. Была реализована модификация Коши, которая замедляет уменьшение температуры, что позволяет рассмотреть больше решений и находить другие более оптимальные маршруты.

Источники

1. PyQt5 documentation // PyQt5 URL: https://pypi.org/project/PyQt5/ (дата обращения: 28.03.2025).
2. Matplotlib documentation // Matplotlib URL: https://matplotlib.org/stable/index.html (дата обращения: 5.03.2025)
3. Networkx documentation // Networkx URL: https://networkx.org/ (дата обращения: 5.03.2025)