Санкт-Петербургский государственный университет
Факультет прикладной математики и процессов управления

Иследование принципов работы метода ближайших соседей

Лабараторная работа №4

Аннотация

В данной работе представлено описание принципов работы алгоритма ближайшего соседа в задаче коммивояжёра, подкреплённое програмной и графической реализацией.

Ключевые слова: Метод ближайшего соседа, Python, Tkinter

Автор работы: Шайдуров В.Д.
Группа: 21.Б15-пу
Научный руководитель: Дик А.Г.

Санкт-Петербург
2023 г.

Содержание

1 Вступление

В данной лабораторной работе мы будем изучать метод ближайшего соседа в задаче о коммивояжере. Эта задача является классической проблемой комбинаторной оптимизации, заключающейся в нахождении кратчайшего пути, проходящего через все заданные точки и возвращающегося в начальную точку. Метод ближайшего соседа - это один из простейших алгоритмов для решения этой проблемы, который основан на принципе выбора следующей точки из всех еще не посещенных точек, ближайшей к текущей позиции.

2 Цель работы

Целью данной лабораторной работы является изучение принципа работы метода ближайшего соседа, а также составление графического интерфейса данного алгоритма для пользователя.

3 Задача

Оценить эффективность метода ближайшего соседа и реализовать его визуализацию.

4 Описание програмы

В данном разделе приводиться описание кода на уровне идеи. Для более подробного понимания, рекомендуется самостоятельно ознакомиться с кодом по ссылке (github). Каждое дейсвие в программе сопровожденно коментариями, если возникают вопросы по терменалогии или по общему устройству программы обращайтесь к этому разделу.

Програмная реализация написана на языке python 3.10, с использованием популярных и общеупотребимых пакетов.

4.1 Теоретические сведения

Необходимые термины:

Верщины графа - есть точки, которые мы понимаем как объекты до которых хотим добраться.

Ребро - есть путь соеденяющий любые две вершины графа.

Гамильтонов цикл - такой замкнутый путь, который проходит через каждую вершину данного графа ровно по одному разу; то есть простой цикл, в который входят все вершины графа.

Примичание: основная цель задачи о коммивояжёре заключается в нахождении минимальноого Гамильтонова цикла.

4.2 Общий ход программы

1. Вызывается класс *Neighbor* в который передаются, веса путей полного графа.
2. Далее через созданный класс вызываем метод *findbestway* который будет начинать обход поочередно с каждой из N вершин графа и на каждом последующем шаге будет выбирать наименьший из возможных путей, когда все вершины будут пройдены, мы стравниваем длины N получившихся маршрутов, т.к. изночально можно было начать с любой из N вершин, и выбираем наименьшую длину пути.

4.3 Описание програмной структуры генетического алгоритма

Программа состоит из одного класса Neighbor описывабщего структуру алгоритма и одного метода findbestway который ищет наилучший, т.е. наименьший маршрут.

В Листиге 1 привёдн испольняемый код для класса Neighbor.

```python
class Neighbor:
    """ Метод ближайшего соседа """
    def __init__(self, graphPaths):
        self.graphPaths = graphPaths  # веса путей заданных в графе
        self.vertexes = len(graphPaths) + 1  # кол-во вершин
        self.best_path = []  # лучший маршрут
        self.best_score = float('inf')  # наименьшая длина цикла

    def find_best_way(self) -> None:
        # Начинаем обход по каждой вершине
        for i in range(self.vertexes):
            new_path = [i]
            new_score = 0
            while len(new_path) <= self.vertexes:
                min_rib = float('inf')
                next_top = i
                for j in range(self.vertexes):
                    if j not in new_path:
                        from_ = min(new_path[-1], j)
                        to_ = max(new_path[-1], j) - from_ - 1
                        rib = self.graphPaths[from_][to_]
                        if rib < min_rib:
                            min_rib = rib
                            next_top = j

                if min_rib < float('inf'):
                    new_score += min_rib
                else:
                    from_ = min(new_path[-1], next_top)
                    to_ = max(new_path[-1], next_top) - from_ - 1
                    new_score += self.graphPaths[from_][to_]
                new_path.append(next_top)

            if new_score < self.best_score:
                self.best_path = new_path
                self.best_score = new_score
```

Listing 1: class Neighbor

5 Визуализация работы программы

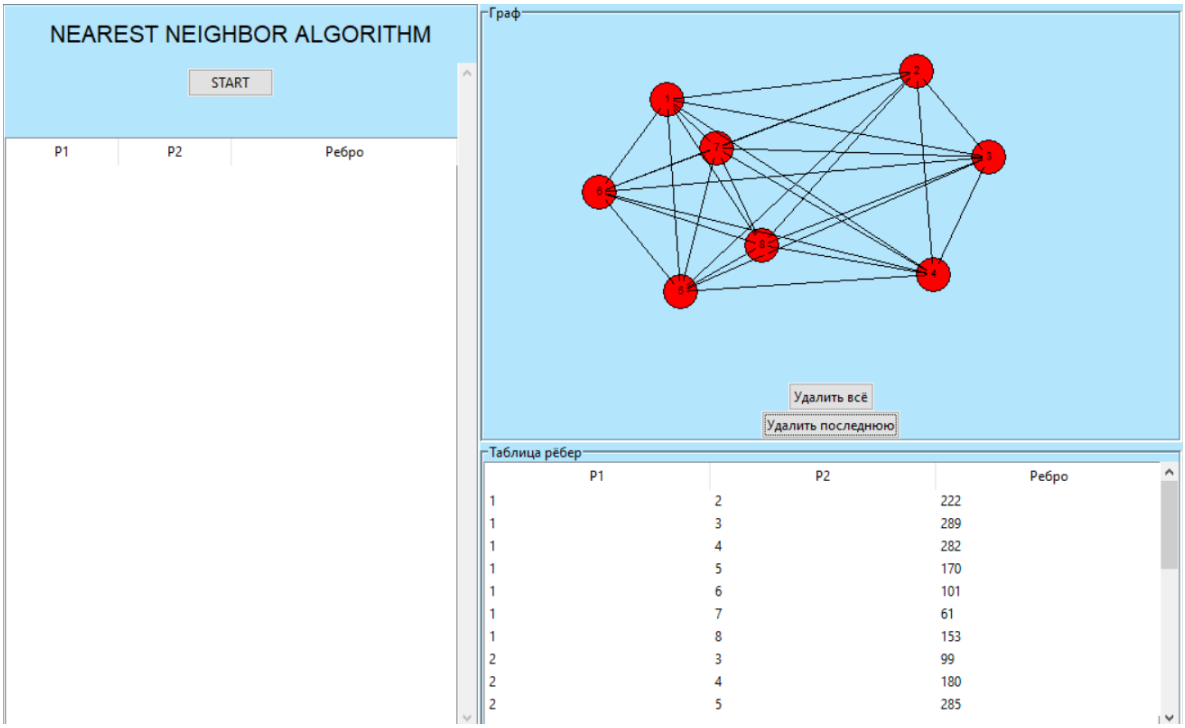

Рис. 1: Окно ввода.

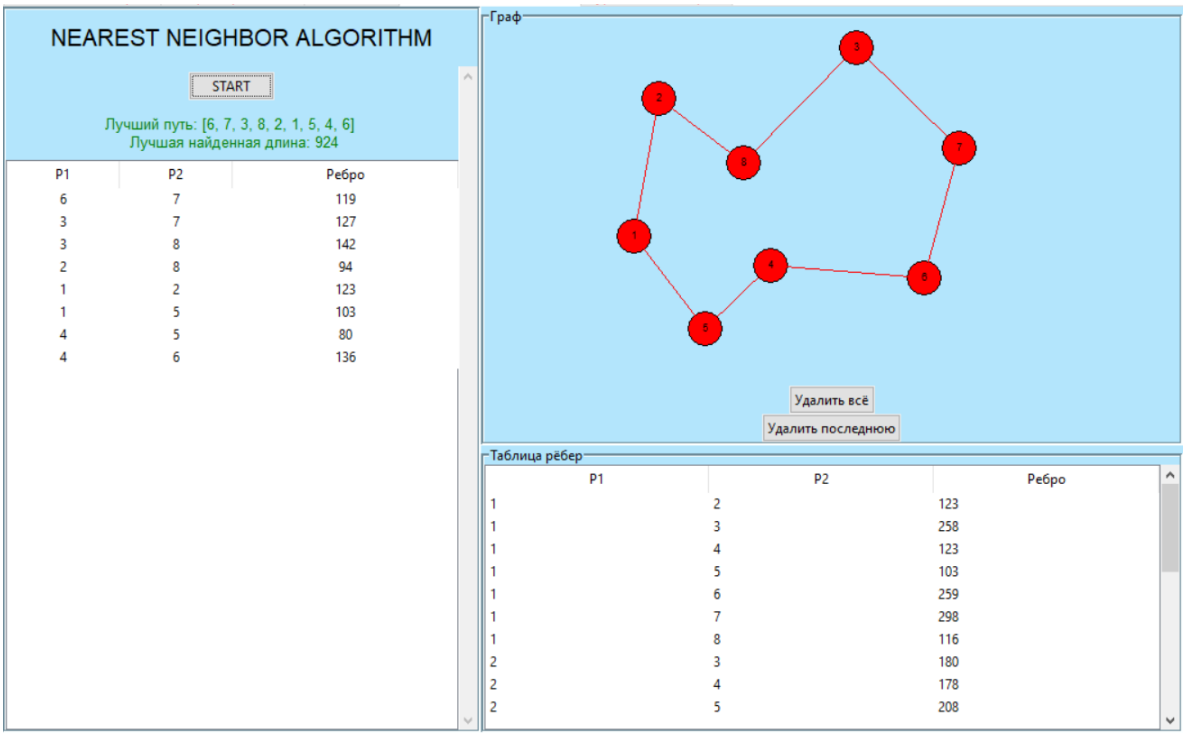

Рис. 2: Окно вывода.

Устройсво приложения:

Вверху справа графическое интерактивное окно, в котором можно добавлять точки одним щелчком мыши. Пути графа рисуются автоматически и добавляются в таблицу рёбер ниже. Весом ребра мо умолчанию является его длина в пикселях, которую можно легко изменить двойным нажатием мыши по нужной ячейке.

Нажимая на кнопку старт, производяться рассчёты, результат которых выводиться под кнопкой зелёным цветом, а таблица заполняется маршрутом в указанном алгоритмом порядке.

6 Оценка эффективности алгоритма

В результате выполнения лабораторной работы мы изучили метод ближайшего соседа в задаче о коммивояжере и реализовали его на примере нескольких задач разной сложности. Метод показал хорошую эффективность для небольших графов, но при увеличении количества вершин или ребер в центра графа он начинает проявлять свои недостатки: оптимальный маршрут может быть найден не всегда. Таким образом, можно заключить, что метод ближайшего соседа - достаточно простой и быстрый алгоритм для решения задачи коммивояжера, который может быть эффективен для малых графов, но для более крупных проявляет свою недостаточную точность.

Заключение

В данной работе был расмотрен алгоритм ближайших соседей на поиск минимального Гамильтона цикла. Были выявленны его недостатки, а также представленна его прогрманая визуализация и теоретическое растолкование.