Метрическая задача коммивояжёра - 2

Мельникова Татьяна, Б05-826

Ноябрь 2020

Содержание

1 Введение

В статье будет рассмотрена метрическая задача коммивояжёра, которая заключается в том, чтобы найти гамильтонов путь минимальной стоимости между двумя фиксированными вершинами во взвешенном графе. Данная задача принадлежит классу NP-полных задач и при большом числе вершин не может быть решена перебором, поэтому необходимо использовать раздичные алгоритмы оптимизации.
Существует множество алгоритмов, позволяющих найти приблежённое решение. Долгое время наилучшим результатом считался алгоритм Кристофидеса [1] дающий $\frac{3}{2}$ - приближение. В 2020 году был разработан немного улучшенный алгоритм апроксимации [2], дающий приближение $\frac{3}{2} - \varepsilon$ для некоторых $\varepsilon > 10^{-36}$
Мы же рассмотрим модификацию алгоритма Кристофидеса, дающую $\frac{5}{3}$ - приближение.[3]

[1] https://clck.ru/RwEAZ
[2] https://arxiv.org/abs/2007.01409
[3] https://clck.ru/RwEiF

2 NP-полнота

2.1 Постановка задачи

Дан граф $G = G(V, E)$ и веса на рёбрах $\omega : V \to \mathbf{R}_+$. Требуется найти гамильтонов путь между двумя фиксированными вершинами минимального веса.

Мы будем решать **метрическую** задачу, т.е. $G(V, E)$ - полный, а для функции весов выполнено неравенство треугольника.

2.2 Доказательство NP-полноты

Теорема: Язык TSP =$(G, w, s, t)|$ во взвешенном графе (G,w) есть путь коммивояжера из s в t длины не более l – **NP**-полный.

Доказательство: Знаем, что язык $UHAMPATH$ - **NP**-полный. Сведем задачу $UHAMPATH$ к TSP. Пусть дан взвешенный граф $G = (V, E)$ с n вершинами. Сопоставим каждому ребру в G единичные веса, и положим $l = n - 1$ (Если $s = t$, возьмем $l = n$). Действительно, путь из $n - 1$ ребра, проходящий через все вершины, обязан быть гамильтоновым.

Замечание: Ясно, что если в формулировке задачи коммивояжёра требуется полнота графа, то это не влияет на суть задачи: все отсутствующие рёбра можно провести, но сопоставить им очень большие веса, так чтобы оптимальный путь через них заведомо не проходил.

В рассуждении об **NP**-полноте можно взять все рёбра, отсутствующие в исходном графе, с весом больше 1: на вывод это не повлияет. При этом веса рёбер образуют метрику, соответственно, метрическая задача коммивояжёра также будет **NP**-полной.

3 Приблежённое решение задачи

3.1 Наивный алгоритм

Пребираем все возможные пути. Асимптотика: $O((n-1)!)$

```
def naive_algo(G, start, finish):
    global min_len
    global minpath
    start = 1
    finish = 4
    minpath = []
    path = [start]
    min_len = 1000000
    n = len(G.nodes())

    def find_path(vertex, prev_vertices, old_len):
        """


        """
        global min_len
        global minpath
        for next_vertex in G.neighbors(vertex):
            if next_vertex not in prev_vertices:
                current_len = old_len
                current_len += G[vertex][next_vertex]['weight']
                path = prev_vertices.copy()
                path.append(next_vertex)

                if (next_vertex == finish and len(path) == n):

                    if (current_len < min_len):
                        minpath = path
                        min_len = current_len
                else:
                    find_path(next_vertex, path, current_len)
        return (minpath, min_len)
    return find_path(start, path, 0)
```

3.2 Алгоритм на основе остовного дерева и паросочетаний, дающий $\frac{5}{3}$ - приближение

Алгоритм, дающий $\frac{5}{3}$- приближение:

1. Построим минимальное остовное дерево MST, с помощью алгоритма Крускала

2. Формируем список вершин W, содержащий вершины, четность которых нужно изменить. Т.е. такие вершины, что

   (a) $v \neq s$ и $v \neq t$ и степень v - нечетная

   (b) $(v == s$ или $v == y)$ и степень v - четная

3. Строим совершенное паросочетание $Match$ на подграфе, содержащем вершины W, с помощью алгоритма Эдмонса

4. Построим граф G‘, состоящий из ребер из MST U $Match$. Заметим, что полученный граф будет *эйлеровым*[4], начало – наша вершина s, а конец – наша вершина t. Полученный граф назовем $EulerGraph$

5. Совершим обход по $EulerGraph$. Путь, который мы найдем, обозначим $EulerPath$.

6. Сконструируем гамильтонов цикл $AlgoHamPath$ по вершинам исходного графа G в порядке, в котором они встречаются в списке вершин $EulerPath$

7. Вернем полученный список $AlgoHamPath$

3.3 Доказательство оценки

Доказательство:

Пусть оптимальный путь – $OptimalHamPath$. Функция $weight(Path)$ возвращает вес пути $Path$. Пространство евклидово, значит

$$weight(AlgoHamPath) \leq weight(MST) + weight(Match)$$

Докажем следующие 2 неравенства:

- $weight(MST) \leq weight(OptimalHamPath)$
- $weight(Match) \leq \frac{2}{3} weight(OptimalHamPath)$

1. $weight(MST) \leq weight(OptimalHamPath)$
   Доказательство:
   Неравенство выполняется по определению минимального остовного дерева. $OptimalHamPath$ - это остовное дерево (считаем, что $s \neq t$, если $s == t$, то разделим эту вершину на 2 вершины и добавим между ними ребро очень большого веса). MST - минимальное остовное дерево, а значит $weight(MST) \leq weight(OptimalHamPath)$

2. $weight(Match) \leq \frac{2}{3} weight(OptimalHamPath)$
   Доказательство:
   Пусть E_{st} - путь между s и t в MST. Покажем, что $weight(MST \backslash E_{st}) \geq weight(Match)$:

   Заметим, что $MST \backslash E_{st}$ можно представить как набор ребер непересекающихся путей между всеми нечетными вершинами в W. Заменим

[4]– в графе G‘ окажутся две вершины нечетной степени по построению (пункты 1-3)

все пути ребром, соедениющим начатьльную вершину пути и конечную. Получим совершенное паросочетание $M^{'}$ на W. $weight(M^{'}) \leq weight(MST \backslash E_{st})$ Т.к. $Match$ - минимальное совершенное пасроочетание на W, то $weight(Match) \leq weight(M^{'}) \leq weight(MST \backslash E_{st})$.

Также Шмойс и Уильямсон[5] показали, что
$2weight(Match) \leq weight(OptimalHamPath)$

С учетом данных неравенств получим:

$$2weight(OptimalHamPath) \geq weight(OptimalHamPath) + weight(MST) =$$

$$= weight(OptimalHamPath) + weight(MST \backslash E_{st}) + weight(E_{st}) \geq$$

$$\geq weight(OptimalHamPath) + weight(Match) + weight(E_{st}) \geq$$

$$\geq weight(OptimalHamPath) + weight(Match) \geq$$

$$2weight(Match) + weight(Match) = 3weight(Match)$$

Итого:

$$weight(AlgoHamPath) \leq weight(MST) + weight(Match) \leq$$

$$\leq weight(OptimalHamPath) + \frac{2}{3}weight(OptimalHamPath) =$$

$$= \frac{5}{3}weight(OptimalHamPath)$$

Что и требовалось показать.

3.4 Реализация алгоритма

(Также реализацию можно посмотреть в прикрепленном ноутбуке tsp.ipynb)

1. Используемые библиотеки

```python
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import networkx.algorithms.approximation
import networkx.algorithms.euler
from mpl_toolkits.basemap import Basemap
```

2. Будем строить гамильтонов путь для городов России с населением $>=$ 100.000

[5]Shmoys, D. B., Williamson, D. P., September 1990. Analyzing the Held-Karp TSP bound: a monotonicity property with application. Information Processing Letters, Vol 35 , Issue 6, Pages: 281 - 285

```
russia_incidents = pd.read_csv('ru.csv')
russia_incidents = russia_incidents[russia_incidents['population'] >= 100000]

Coord_X = list(russia_incidents.lng)
Coord_Y = list(russia_incidents.lat)
```

3. Строим граф

```
G=nx.Graph()
for i in range(len(Coord_X)):
    for j in range(len(Coord_Y)):
        G.add_edge(i,j,weight=np.sqrt((float(Coord_X[i])-float(Coord_X[j]))**2
                    + (float(Coord_Y[i])-float(Coord_Y[j]))**2))
```

4. Посмотрим минимальное остовное дерево MST (*см. рис.1*). В данной задаче будем использовать алгоритм Крускала Этот алгоритм работает за полиномиальное время от $|V|$.

```
MST = nx.minimum_spanning_tree(G)
draw_graph(MST, Coord_X, Coord_Y, "mst.svg")
```

Рис. 1:

5. Формируем список вершин W, содержащий вершины, четность которых нужно изменить. Т.е. такие вершины, что

   (a) $v \neq s$ и $v \neq t$ и степень v - нечетная

   (b) ($v == s$ или $v == y$) и степень v - четная

```
W = []
for node in MST.nodes() :
    if (MST.degree[node] % 2 != 0 and not(node==s or node==t)):
        W.append(node)

if (MST.degree[s] % 2 == 0):
        W.append(s)
if (MST.degree[t] % 2 == 0):
        W.append(t)
```

6. Строим совершенное паросочетание Match на подграфе G, содержащем вершины W, с помощью алгоритма Эдмонса.

```
sub_G = nx.subgraph(G, W).copy()

for u,v,d in sub_G.edges(data=True):
    if d['weight'] != 0:
        d['weight'] = 1/d['weight']
Match = nx.max_weight_matching(sub_G)
```

7. Теперь составим новый граф G‘, состоящий из ребер из MST U Match. Заметим, что полученный граф будет эйлеровым. Полученный граф назовем EulerGraph (*см. рис.2*)

```
union = Match
for edge in MST.edges():
    union.append(edge)

EulerGraph = nx.MultiGraph()
EulerGraph.add_edges_from(union)
```

Рис. 2:

8. Совершим обход по EulerGraph. Путь, который мы найдем, обозначим EulerPath.

```
EP = [edge for edge in nx.eulerian_path(EulerGraph, source=
    start)]
EP_nodes = []
for edge in EP:
    EP_nodes.append(edge[0])
EP_nodes.append(EP[-1][1])
```

9. Сконструируем гамильтонов цикл AlgoHamPath по вершинам исходного графа G в порядке, в котором они встречаются в списке вершин EulerPath (*см. рис.3*)

```
AlgoHamPath = []
for node in EP_nodes:
    if not(node in AlgoHamPath) and node != EP_nodes[-1]:
        AlgoHamPath.append(node)

AlgoHamPath.append(EP_nodes[-1])
```

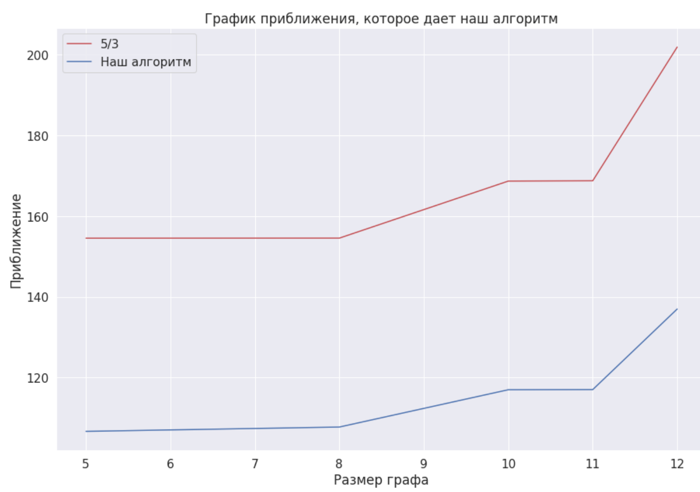
Рис. 3:

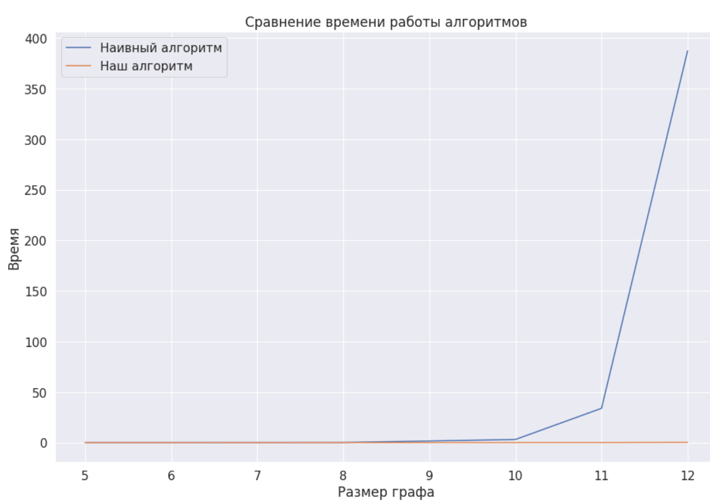
4 Результаты тестов

4.1 Описание тестов

Посмотрим как ведут себя алгоритмы на графах размера [5, 8, 10, 11, 12]. Граф будем строить на основе датасета с городами России[6]

4.2 Анализ результатов





[6]https://simplemaps.com/data/ru-cities

По полученным графикам видно, что уже при 11 вершинах время работы наивного алгоритма сильно превосходит время работы приблежённого алгоритма. Также выплнена оценка: $weight(AlgoHamPath) \leq \frac{5}{3} weight(OptimalHamPath)$, что говорит о корректности нашего алгоритма.

5 Выводы

Задачи, связанные с TSP, сегодня не имеют оптимального решения. Ученые годами ищут новые алгоритмы, позволяющие улучшить приближение. Последнее достижение произошло в 2020 году[7]. Решение этой задачи очень важно – оно носит не только теоретический, но и прикладной характер.

[7] https://arxiv.org/abs/2007.01409

6 Литература

1. https://habrahabr.ru/post/125898/
2. http://e-maxx.ru/index.php
3. https://arxiv.org/abs/1310.1896
4. J. Edmonds. Path, trees, and flowers. Canadian J. Math., 17:449–467, 1965.
5. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
6. http://www.mscs.dal.ca/ janssen/4115/presentations/Poppy.pdf
7. http://mathworld.wolfram.com/EulerianCycle.html