«Задача о деревенском почтальоне (RURAL POSTMAN, RPP)»

ФПМИ МФТИ

Иванов Максим Олегович

Лето 2024

Аннотация

Задача о деревенском почтальоне заключается в том, чтобы в графе с некоторым подмножеством рёбер найти цикл минимального суммарного веса, хотя бы один раз проходящий через каждое ребро из данного подмножества. В этой статье будет доказано, что эта задача является **NP-полной**, будет доказана полиномиальная разрешимость в особом случае графа, а также имплементирован алгоритм для решения задачи в этом случае.

Содержание

Введение

Формулировка задачи

Задача о деревенском почтальоне (RURALPOSTMAN, RPP) заключается в том, чтобы в графе с некоторым подмножеством рёбер найти цикл минимального суммарного веса, хотя бы один раз проходящий через каждое ребро из данного подмножества.

Более формально: дан граф $G = (V, E)$, веса рёбер $w : E \to \mathbb{R}_+$ и множество рёбер $R \subset E$. Требуется найти цикл минимального суммарного веса, хотя бы один раз проходящий через каждое ребро из R.

Несмотря на то, что эта задача является **NP-полной**, при наличии дополнительных условий на структуру графа RPP разрешима за полиномиальное время: граф, образованный рёбрами R, должен иметь константное число компонент связности. Эти факты будут доказаны в статье далее. Также будет приведён алгоритм для случая с полиномиальной разрешимостью.

История задачи и значение результатов

Задача RPP является одной из задач множества ARP (Arc Routing Problems). Это задачи, в которых один или несколько «исполнителей» должны выполнить задачи для некоторых «заказчиков», смоделированных в виде ребер графа, и оптимизировать достижение заданной цели. [1] Например, в нашем случае «заказчики» — улицы (подмножество рёбер), по которым почтальон разносит письма. Оптимизация заключается в минимизации стоимости цикла.

Задача о деревенском почтальоне, введённая К.Орловым — обобщение хорошо известной задачи китайского почтальона (CPP), где множества R и E совпадают. Данная постановка задачи была введена М. Квон. В начале 1970-х исследованием этой задачи занимались учёные из США, Канады и СССР [2].

К реальным приложениям RPP относятся: маршрутизация транспортных средств для доставки газет или почты, маршрутизация снегоуборочных машин и школьных автобусов или проверка линий электропередач. [3]

Также есть различные модификации задачи о деревенском почтальоне. Приведём некоторые из них:

1. *Задача о деревенском почтальоне с определённым количеством транспортных средств и с органичением на длину маршрута (Length Constrained K–vehicles Rural Postman Problem,* LC K − RPP*)* [1]
2. *Задача о деревенском почтальоне с определённым количеством транспортных средств (K-vehicles Rural Postman Problem,* K − RPP*)* [1]

Задачи, связанные с использованием нескольких транспортных средств, гораздо сложнее решить, и для их решения требуются условия, позволяющие сбалансировать протяженность маршрутов различных транспортных средств. Широко используемым условием является ограничение длины маршрута каждого транспортного средства определенной величиной (пункт 1).

3. *Задача о деревенском почтальоне на неориентированном графе (Undirected Rural Postman Problem,* URPP*)*

4. *Задача о деревенском почтальоне с проверкой перекрёстка (Intersection Inspection Rural Postman Problem,* IIRPP*)* [4]

   Можно привести пример, когда эта задача применима: местные органы власти инспектируют дороги, чтобы решить, какие участки следует отремонтировать. Инспекционный автомобиль выбирает направление движения на перекрёстке, чтобы полностью охватить участок. Требуется оптимизировать пройденное расстояние.

Известные результаты

Существуют несколько типов алгоритмов, решающих RPP:

1. *Метаэвристические алгоритмы*

   Цель метаэвристики состоит в эффективном исследовании пространства поиска для нахождения (почти) оптимальных решений. [5] Например, Ant Colony Optimization (ACO) [7] — это метаэвристический алгоритм, который решает задачу RPP.

2. *Эвристические алгоритмы*

   Это алгоритмы решения задачи, включающие практические методы, не являющиеся гарантированно точными или оптимальными, но достаточные для решения поставленной задачи. [6] Например, для нашей задачи это алгоритм Кристофидеса. [3]

3. *Точные алгоритмы*

   Алгоритмы используются для поиска точных решений с помощью математических методов, однако они могут быть затратными для вычисления. Например, алгоритм ветвления и привязки для решения задачи RPP [8] — это точный алгоритм.

Далее в статье мы реализуем точный алгоритм для решения задачи RPP в случае, когда граф, образованный рёбрами R, имеет константное число компонент связности.

Техническая часть

Определения и обозначения

Def. *Взвешенным неориентированным графом* называется пара множеств $G = (V, E)$ с функцией w, где V — множество каких-то объектов, E — множество пар объектов из V, а $w : E \to \mathbb{R}_+$ — функция стоимости рёбер (считаем, что стоимости положительны).

Note. Мы допускаем, что граф может быть непростым, то есть в нём могут содержаться петли и кратные рёбра.

Def. *Циклом в графе* называется замкнутый обход графа.

Def. *Мультимножеством* называется структура со свойствами множества, допускающая включение одного и того же элемента в совокупность по нескольку раз.

Def. *Задача коммивояжёра* (TSP) заключается в поиске самого выгодного обхода графа, проходящего через указанные вершины хотя бы по одному разу с последующим возвратом в исходную вершину. [9]

Def. *Задача разрешимости* — вопрос, сформулированный в рамках какой-либо формальной системы, требующий ответа «да» или «нет», возможно, зависящего от значений некоторых входных параметров. [10]

Введём вспомогательные обозначения:

Пусть S — некоторое мультимножество рёбер графа $G = (V, E)$. Возьмём $R \subset E$ из формулировки задачи, (G, w, R) — постановка задачи RPP. Обозначим $V(S)$ множество конечных вершин ребер из S, $G(S) = (V(S), S)$. В $G(S)$ нет изолированных вершин, но кратность рёбер может превышать кратность рёбер в G.

Также положим для некоторого $A \subset E$: $w(A) = \sum_{e \in A} w(e)$.

Def. *Эйлерово расширение графа* для постановки (G, w, R) задачи RPP — мультимножество S рёбер графа G, такое что граф $G(R \sqcup S)$ эйлеров, то есть в нём есть цикл, посещающий каждое ребро по одному разу. [2]

Def. *Совершенным паросочетанием* в графе называется набор попарно несмежных рёбер, в котором участвуют все вершины графа. [11]

Вспомогательные утверждения

Лемма (1) (б/д)
С точки зрения задачи разрешимости задача коммивояжёра (TSP) является **NP-полной**. [9]

Note. Далее считаем, что метрическая и общая постановки RPP эквивалентны. [2]

Основная часть

Доказательство NP-полноты задачи

Для доказательства **NP-полноты** RPP потребуется доказать принадлежность RPP классу **NP**, а также то, что можно свести любую другую задачу из **NP** за полиномиальное время (**NP-трудность**).

Теорема (1). Задача RPP принадлежит классу **NP**.

Доказательство:

С точки зрения задачи разрешимости задача RPP звучит так: «существует ли маршрут не длиннее, чем заданное значение k». Тогда в качестве сертификата полиномиальной длины можно взять маршрут почтальона. Задача удовлетворяет определению класса **NP** через сертификаты, а значит, принадлежит ему.

Теорема (2). Задача RPP является **NP-трудной**. [2]

Доказательство:

Сведём к задаче RPP задачу коммивояжёра (по лемме 1 она **NP-полная**). Пусть (G, w) — постановка задачи TSP, где G — полный граф, w удовлетворяет неравенству треугольника.
Построим постановку задачи о деревенском почтальоне. Граф G' получим из G добавлением петель нулевой стоимости, по одной из каждой вершины. R состоит из добавленных петель. Тогда любое решение W для задачи TSP (G, w) индуцирует решение W' для задачи RPP (G', w', R) добавлением к W петель из R. При этом $w'(W') \leqslant w(W)$.
С другой стороны, любое решение W' для задачи RPP (G', w', R) индуцирует решение W для задачи TSP (G, w) сокращением повторных посещений вершин. При этом $w(W) \leqslant w'(W')$.
Следовательно, стоимости решений совпадают, а значит, полиномиальное сведение построено. [2]

Доказательство полиномиальной разрешимости задачи с дополнительным условием

Построим алгоритм для RPP, который работает за полиномиальное время при константном числе компонент связности в графе. Ограничимся следующими свойствами постановки (G, w, R):

- Граф $G = (V, E)$ полный и $V = V(R)$. Тогда любое решение задачи приходит по всем вершинам графа.
- w удовлетворяет неравенству треугольника.

Note. Обозначим через c число компонент связности в графе $G(R)$.

Лемма (2)

Пусть (G, w, R) — постановка задачи RPP, S — эйлерово расширение для такой постановки, что нет эйлерова расширения S' меньшей мощности и стоимости. Тогда $S = T \sqcup M$, где $T \subset S$ — $(c-1)$ ребро и $G(R \sqcup T)$ — связный граф, $M \subset S$ — совершенное паросочетание на вершинах нечётной степени в $G(R \sqcup T)$. [2]

Доказательство:

Разобьём $S = T \sqcup M$ так, что $T \subset S$ — минимальное по включению множество рёбер такое, что $G(R \sqcup T)$ — связный граф, мощность множества равна $(c-1)$.
Докажем, что M — паросочетание. Предположим противное и найдём 2 ребра: $\{u, v\}, \{v, w\} \in M$. Рассмотрим множество $S' = (S \setminus \{u, v\}, \{v, w\}) \sqcup \{u, w\}$. Очевидно, что $|S'| < |S|$ и по неравенству треугольника $w(S') \leqslant w(S)$. Чётности всех вершин в $G(R \sqcup S)$ и в $G(R \sqcup S')$ совпадают. Кроме того, $G(R \sqcup S')$ связен, так как $T \subset S'$. Тогда S' также является эйлеровым расширением для постановки задачи – получаем противоречие к выбору множества S. [2]

Теорема (3). Задача RPP решается за время $O(n^{2c-2} \cdot n^3)$ при числе компонент связности в графе равном $c = const$. [2]

Доказательство:

Будем перебирать всевозможные множества T из *Леммы (2)*. Выбираем 2 конечные вершины для каждого из $(c-1)$ ребра. Тогда таких множеств будет $O(n^{2c-2})$.
Для каждого множества T вычисляем совершенное паросочетание минимального веса M на вершинах нечётной степени в $G(R \sqcup T)$ за время $O(n^3)$. Из всех вариантов $R \sqcup T \sqcup M$ выбираем минимальный по стоимости. [2]

Имплементация алгоритма

Итак, опишем алгоритм, удовлетворяющий *Лемме (2)* и *Теореме (3)*:

1. Ищем все компоненты связности в графе. Их константное число.
2. Перебираем всевозможные множества рёбер T.
3. Для каждого множества T вычисляем совершенное паросочетание минимального веса.
4. Строим цикл, содержащий все рёбра из R и T. Если его стоимость минимальная, обновляем минимальный цикл.

Код написан на языке программирования Python с использованием библиотек `networkx` для работы с графами и `itertools` для перебора множеств T.

Код проекта на есть в репозитории на GitHub.

```python
import networkx as nx
import itertools

def rpp(graph, required_edges):
    weights = nx.get_edge_attributes(graph, 'weight')
    # Поиск всех компонент связности в графе
    components = list(nx.connected_components(graph))
    num_components = len(components)

    # Цикл для перебора всех возможных подмножеств рёбер T
    min_cycle_cost = float('inf')
    optimal_cycle = None
    for component in components:
        for T in itertools.combinations(component, num_components-1):
            G = graph.copy()
            G.add_edges_from(T)

            # Поиск паросочетания на вершинах нечётной степени
            odd_degree_nodes = [node for node in G.nodes() if G.degree(node) % 2 != 0]
            odd_degree_subgraph = G.subgraph(odd_degree_nodes)
            matching = nx.max_weight_matching(odd_degree_subgraph, weight='weight')

            # Построение цикла, содержащего все рёбра из R и T
            cycle_edges = required_edges.union(set(T)).union(set(matching))
            cycle_cost = 0
            for edge in cycle_edges:
                if edge in weights:
                    cycle_cost += weights[edge]
                else:
                    cycle_cost += weights[(edge[1], edge[0])]

            # Обновление оптимального цикла
            if cycle_cost < min_cycle_cost:
                min_cycle_cost = cycle_cost
                optimal_cycle = cycle_edges

    return optimal_cycle, min_cycle_cost
```

Функция `rpp(graph, required_edges)` принимает на вход граф G и множество R и возвращает

множество рёбер минимального цикла и стоимость этого цикла.

Пример использования можно посмотреть в репозитории на GitHub.

Тестирование алгоритма

Тесты запускаются с помощью `pytest`. Код тестов доступен на GitHub. Наборы тестов были использованы с http://users.cecs.anu.edu.au.

Результаты бенчмарка на разных наборах тестов.

```
-------------------------------------------------------------------------------------
Name (time in us)               Min                    Max                Mean
-------------------------------------------------------------------------------------
test_benchmark_5edges         9.6790 (1.0)          32.0830 (1.0)       10.3422 (1.0)
test_benchmark_10edges      141.6900 (14.64)     9,219.5840 (287.37)   156.0450 (15.09)
test_benchmark_15edges      142.2210 (14.69)       480.3270 (14.97)    154.8035 (14.97)
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
   StdDev               Median               IQR            Outliers  OPS (Kops/s)            Rounds  Iterations
-------------------------------------------------------------------------------------------------------------------
   0.9762 (1.0)        10.0170 (1.0)       0.9273 (1.0)       637;260       96.6915 (1.0)       19233           1
 143.0253 (146.51)   145.5460 (14.53)     5.2960 (5.71)        4;791        6.4084 (0.07)       4110           1
  21.6869 (22.22)    146.8225 (14.66)     4.8200 (5.20)      567;810        6.4598 (0.07)       4874           1
```

Как мы видим, при увеличении количества рёбер время выполнения алгоритма кратно увеличивается, однако алгоритм работает за полиномиальное время, как было доказано ранее.

Список литературы

Список литературы

[1] Angel Corberan, Isaac Plana, Jose M. Sanchis, Paula Segura (2021) *Polyhedral study of a new formulation for the Rural Postman Problem*

[2] Е. Д. Незнахина (2022) *Алгоритмы с оценками для задач маршрутизации*

[3] W.L. Pearn, T.C. Wu (1995) *Algorithms for the rural postman problem*

[4] Debdatta Sinha Roy, Adriano Masone, Bruce Golden, Edward Wasil (2020) *Modeling and Solving the Intersection Inspection Rural Postman Problem*

[5] Щербина О.А. (2014) *Метаэвристические алгоритмы для задач комбинаторной оптимизации (обзор)*

[6] Википедия *Эвристический алгоритм*

[7] M. L. Pérez-Delgado (2007) *A Solution to the Rural Postman Problem Based on Artificial Ant Colonies*

[8] Nicos Christofides, V. Campos, Ángel Corberán, E. Mota (1981) *An algorithm for the Rural Postman Problem*

[9] Википедия *Задача коммивояжёра*

[10] Википедия *Задача разрешимости*

[11] Википедия *Паросочетание*