Московский физико-технический институт (ГУ)
Физтех-школа прикладной математики и информатики
Сложность вычислений, осень 2023

Задача о деревенском почтальоне. Полиномиальный алгоритм при ограничениях и его анализ.

Крехов Николай
Б05-127

Москва, 2023

Аннотация

В этом проекте будет рассмотрена задача о деревенском почтальоне (Rural Postman Problem - RPP), которая является фундаментальной в классе комбинаторных оптимизаций и регулярно появляется в прикладных задачах. Автор провел исследование этой **NP**-полной задачи. Обоснован и реализован полиномиальный алгоритм ее решения при некоторых ограничениях на граф.

Содержание

1 Введение

Задача деревенского почтальона (RPP) является расширением более известной задачи о китайском почтальоне (Chinese Postman Problem - CPP), в которой необходимо найти цикл минимальной стоимости, проходящий через все ребра. В RPP же задается множество необходимых для посещения ребер. Задача имеет прикладное значение, например, для доставки почты, сборки мусора вдоль улиц, проверки на исправность трубопровода и для многих других. Эти задачи, как и задача о коммивояжере (TSP), являются частными случаями более общей задачи Vehicle Routing Problem [1].

Впервые RPP была выдвинута C. S. Orloff [1], и сразу привлекла к себе интерес ученых. В общем случаем RPP является **NP**-полной, к ней сводится задача о коммивояжере. Однако при некоторм ограничени на множество ребер, которые необходимо посетить, существует полиномиальный алгоритм поиска решения. Это ограничение заключается в том, что количество компонент связности графа на множестве ребер, которое необходимо посетить является фиксированной константой $c > 1$, то есть этот граф не является связным. Тогда время работы алгоритма будет составлять $O(n^{2c+1})$.

2 Техническая часть

2.1 Постановка задачи

Определение 1. ***Задача о деревенском почтальоне (RPP)***
Дан граф $G = (V, E)$, веса рёбер $w : E \to N$, и множество рёбер $R \subset E$. Требуется найти цикл минимального суммарного веса, хотя бы один раз проходящий через каждое ребро из R.

Утверждение 1. *Поставленная задача* **NP***-полна.*

Теорема 1. *Дана задача $RPP(G, w, R)$. Пусть $G\langle R\rangle$ - граф, построенный на ребрах R, $c > 0$ - число компонент связности в $G\langle R\rangle$. Тогда задача о сельском почтальоне решается за $O(n^{2c-2} \cdot n^3)$.*

2.2 Вспомогательные определения и утверждения

Замечание к определению 1.
В случае, если множество необходимых для посещения ребер R совпадает со всеми ребрами графа G, задача называется *задачей о китайском почтальоне (CPP).*

Определение 2. ***Задача о коммивояжере (TCP)***
Дан граф $G = (V, E)$ и веса на рёбрах $w : E \to R_+$. Требуется найти гамильтонов цикл минимального веса.

Замечание к определению 2.
В случае, если граф G полный, а функция весов метрическая (то есть для любых трёх вершин x, y и z выполнено $w(x, z) \leq w(x, y) + w(y, z)$), то задача называется *метрической задачей о коммивояжере.*

3 Основная часть

3.1 Доказательство NP-полноты задачи о деревенском почтальоне (Утверждение 1)

Поставленная задача может быть переформулирована в задачу принятия решения следующим образом:
Пусть в дополнения к предыдущим входным данным дается некоторое положительное число K. Существует ли решение исходной задачи при условии, что стоимость цикла должна быть меньше K?

Принадлежность к классу NP. RPP $\in$ **NP**, так как недетерминированная машина Тьюринга может за полиномиальное время проверить, существует ли проходящий по ребрам из R цикл, сумма весов которого меньше K.

NP-трудность. К задаче RPP сводится задача коммивояжёра (TCP из Определения 2). Пусть (G, w) — пример задачи TCP, где G — граф и w - стоимости рёбер. Сведём её к RPP (G', w', R): граф G' получается из G добавлением петли нулевой стоимости к каждой вершине. Множество R состоит из добавленных петель. Пусть W - решение TCP (G, w), тогда $W' = W \sqcup R$ - решение для RPP (G', w', R), так как путь будет содержать R, а петли не добавят стоимости, поэтому цикл будет минимальной стоимости. С другой стороны, если W не является решением TCP (G, w), то $W' = W \sqcup R$ - не будет являться решением для RPP (G', w', R). Таким образом, **NP**-полнота задачи о деревенском почтальоне доказана.

3.2 Алгоритм решения задачи и его анализ

Заметим, что RPP(G', w', R) сводится к (G, w, R), где G - полный граф, w - метрическая функция весов, то есть для любых трёх вершин x, y и z выполнено $w(x, z) \leq w(x, y) + w(y, z)$).
Для этого:

1) Добавим ко всем ребрам графа G' максимальный вес ребра в графе $w'_{max} = max_{x,y} w'(x, y)$. Тогда неравенство треугольника будет выполнено: $w(x, y) = w'(x, y) + w'_{max} \leq 2w'_{max} \leq 2w'_{max} + w'(x, z) + w'(z, y) = w(x, z) + w(z, y)$. Цикл минимальной стоимости при этом останется тем же. Предположим, решение W не совпадает с W', так как ко всем ребрам прибавилось одинаковое значение, это значит, что длина цикла W' отличается от длины цикла W. Пусть $m = |W'|, n = |W|$, $cost(A)$ - суммарная стоимость ребер из множества A с функцией весов w, а $cost'(A)$ - с функцией весов w'. Предположим $m < n$, тогда $cost'(W) = cost(W) - nw'_{max} < cost(W') - mw'_{max} = cost'(W')$, противоречие с тем, что W' - решение исходной задачи. В случае $m > n$ аналогично получим

противоречие.

2) Для пар вершин, между которыми ребро отсутствует, проводим ребро со стоимостью ∞. Тогда они не могут содержаться в решении, так как цикл не будут минимальным.

Лемма 1. *Пусть* $(G = (V, E), w, R)$ *- задача RPP,* $c > 1$ *- число компенент связности в* $G\langle R\rangle$*, а* W *- решение задачи. Тогда* $W = R \sqcup T \sqcup M$*, где* T *- множество из* $c-1$ *ребра такое, что* $G\left\langle R \sqcup T\right\rangle$ *- связный,* M *- совершенное паросочетание на вершинах нечетной степени в* $G\left\langle R \sqcup T\right\rangle$*.*

Доказательство. Разобьем $W \setminus R$ на $T \sqcup M$ так, что T - минимальное по включению множество рёбер, что $G\left\langle R \sqcup T\right\rangle$ - связный. Заметим, что размер множества равен $c-1$, так как достаточно упорядочить компоненты связности и по очереди соединить. Докажем, что M - паросочетание. Пусть нет, тогда найдем два ребра $(u, v), (v, w) \in M$ и заменим их на одно ребро (u, w), ответ не станет хуже по неравенству треугольника. Так как в W нет вершин нечётной степени(так как это цикл), то M - паросочетание на вершинах нечётных степеней из $G\left\langle R \sqcup T\right\rangle$.

Доказательство Теоремы 1. Будем перебирать всевозможные множества T из леммы. Таких не более чем n^{2c-2}, так как для каждого из $c-1$ ребра выбираем по 2 вершины из соответствующих компонент связности, размер которых не превосходит n. Для каждого множества T вычисляем совершенное паросочетание минимального веса M в графе на нечётных вершинах в $G\left\langle R \sqcup T\right\rangle$ за время $O(n^3)$. Из всех перебранных T и M возвращаем решение $R \sqcup M \sqcup T$ минимальной стоимости.

Анализ работы Для приведенного алгоритма

Algorithm 1 Основная функция(Псевдокод)

```
def main(graph, requiredSubgraph):
    #Floyd algorithm to find
    #distance between every pair
    vector<vector<int>> distances = graph.Floyd()
    #degrees of vertices in a graph G<R>
    #constructed on edges from R.
    vector<int> degrees = requiredSubgraph.GetDegrees();
    int minimalCycleWeight = INF
    vector<pair<int, int>> res(
        requiredSubgraph.GetComponentsReference().size()
    );
    # index of connected component
    const int firstComponentIndex = 1;

    # trying to find best set T
    # and perfect matching M by brute force
    chooseVariant(
        requiredSubgraph.GetComponents(),#get connective componen
        dist,
        firstComponentIndex,
        res,
        degrees,
        minimalCycleWeight
    );

    for ([u, v]: requiredSubgraph) {
        minimalCycleWeight += dist[u][v];
    }
    return minimalCycleWeight
```

Algorithm 2 Функция перебора вариантов(Псевдокод)

```
)
def chooseVariant(
    components
    dist,
    firstComponentIndex,
    res,
    degrees,
    minimalCycleWeight
):
    if (firstComponentIndex == components.size()) {
        int cur = 0;
        for ([u, v]: res)
            cur += gr[u][v];
        #Kun's algorythm for min cost matching
        cur += kun(gr, degrees);
        ans = std::min(ans, cur);
        return;
    }
    for (int idx1 = 0;
        idx1 < components[firstComponentIndex].size(); ++idx1) {
        for (int secondComponentIndex = 0; secondComponentIndex
            < firstComponentIndex; ++secondComponentIndex) {
            for (int idx2 = 0; idx2
                < components[compopnentIdx2].size(); ++idx2) {

                int u = components[firstComponentIndex][idx1];
                int v = components[secondComponentIndex][idx2];

                res[firstComponentIndex] = {u, v};

                ++degrees[u];
                ++degrees[v];

                chooseVariant(
                    components,
                    gr,
                    firstComponentIndex + 1,
                    res,
                    degrees,
                    ans
                );

                --degrees[u];
                --degrees[v];
            }
        }
    }
```