78. Задача о деревенском почтальоне-2

ФПМИ МФТИ

Карпов Степан

Содержание

Введение

Постановка задачи

Задача деревенского почтальона (RPP) - это задача теории графов, которая заключается в нахождении цикла минимального суммарного веса, хотя бы один раз проходящего через каждое ребро из заданного множества ребер R. Эта проблема имеет множество практических применений, таких как оптимизация маршрута доставки почты или товаров в разные регионы.

Задача о деревенском почтальоне является продолжением известной задачи о китайском почтальоне (CPP), в которой нужно пересечь подмножество рёбер графа с минимальными затратами.

Решение RPP считается сложной задачей, поскольку известно, что она является NP-трудной, а это означает, что маловероятно, что существует эффективный алгоритм для поиска оптимального решения. Однако были разработаны различные алгоритмы аппроксимации, позволяющие находить хорошие решения для RPP за разумный промежуток времени.

В этой статье мы обсудим различные алгоритмы, которые были предложены для решения RPP, включая их сильные и слабые стороны, и предоставим обзор текущего уровня техники в решении этой проблемы. Мы также рассмотрим некоторые практические применения и влияние, которое он оказывает в реальных сценариях.

Определим формальное условие задачи RPP:

Есть связный граф $G = (V, E)$, где V - множество вершин, а E - множество ребер, где каждое ребро имеет неотрицательный вес w. Также задано некоторое подмножество ребер графа $R \subset E$. Требуется найти цикл минимального суммарного веса (под весом подразумевается сумма весов всех ребер, входящих в данный цикл), хотя бы один раз проходящий через каждое ребро из R.

Существующие исследования

Задача деревенского почтальона хорошо изучена в теории графов. Многие ученые предложили различные алгоритмы и методы для решения RPP, также был проведен ряд исследований, направленных на изучение задачи с различных точек зрения.

Одним из направлений исследований в RPP является сосредоточение внимания на разработке алгоритмов аппроксимации, которые могут находить решения, близкие к оптимальным. Одним из наиболее известных алгоритмов в этой области является алгоритм Кристофидеса, который мы рассмотрим более подробно ниже. Были также предложены другие алгоритмы аппроксимации,

такие как генетические алгоритмы, алгоритмы локального поиска и метаэвристика.

Другое направление исследований сосредоточено на изучении свойств RPP, таких как его сложность и структурные характеристики графа, которые влияют на производительность алгоритмов. Ученые также изучали задачу с точки зрения целочисленного программирования и сформулировали математические модели для ее решения.

Более того, некоторые исследования были проведены в рамках конкретных сценариев или ограничений. Например, задача деревенского почтальона с временными рамками, задача деревенского почтальона с несколькими складами и задача деревенского почтальона с ограниченной вместимостью. Эти исследования показали, что в этих сценариях задача становится еще сложнее.

Наконец, многие ученые пытались применить RPP к реальным сценариям, таким как доставка почты и планирование транспортной сети. Эти исследования показали потенциальные преимущества и проблемы использования RPP в этих областях и привели к разработке новых алгоритмов и методов, адаптированных к конкретным жизненным ситуациям.

Алгоритмы решения

Существует несколько алгоритмов, которые были предложены для решения задачи деревенского почтальона, которые, как упоминалось выше, можно разделить на три категории: алгоритмы аппроксимации, точные алгоритмы и метаэвристики.

1. **Алгоритмы аппроксимации:** одним из наиболее известных алгоритмов аппроксимации для RPP является алгоритм Кристофидеса, который имеет точность $3/2$. Этот алгоритм работает путем построения минимального остовного дерева графа, а затем добавления ребер минимального веса. Полученный мультиграф затем преобразуется в гамильтонов цикл с использованием эйлерового цикла.

2. **Точные алгоритмы:** эти алгоритмы гарантируют нахождение оптимального решения, но они могут быть дорогостоящими с точки зрения вычислений и не способны решить сложные случаи задачи. К ним относятся алгоритмы ветвления и привязки, которые исследуют пространство решений путем систематического устранения неоптимальных решений. Кроме того, точные алгоритмы включают подходы к целочисленному программированию, которые сформулировали математические модели для решения проблемы RPP.

3. **Метаэвристики**: эти алгоритмы находят хорошие решения за разумный промежуток времени, но они не гарантируют нахождения оптимального решения. Примеры метаэвристики включают генетические алгоритмы, имитацию отжига, поиск табу и оптимизацию муравьиной колонии.

Эти алгоритмы могут быть очень эффективными, но они требуют точной настройки параметров и могут плохо масштабироваться для больших графов.

Стоит отметить, что не существует универсального наилучшего алгоритма для решения задачи RPP, выбор алгоритма зависит от конкретных требований и ограничений задачи, а также компромисса между качеством решения и вычислительным временем.

Алгоритм Кристофидеса

Алгоритм Кристофидеса - это аппроксимационный алгоритм для решения задачи деревенского почтальона (RPP), который имеет точность наихудшего случая 3/2. Впервые он был предложен Никосом Кристофидесом.

Никос Кристофидес - греко-канадский специалист по информатике, родившийся в Лимассоле, на острове Кипр, и учившийся в Национальном техническом университете Афин и Имперском колледже Лондона. Он опубликовал первый эффективный алгоритм для решения RPP в 1976 году, и это был первый алгоритм решения задачи за полиномиальное время. Его вклад в RPP был очень существенным в области исследования операций и теории графов. После своей исследовательской карьеры он также сыграл ключевую роль в развитии области вычислительной биологии, изучая такие проблемы, как проблема выравнивания множественных последовательностей и предсказание структуры белка.

Алгоритм Кристофидеса является замечательным вкладом в области теории графов. Он изменил наш подход к проблеме RPP и открыл много дверей для дальнейших исследований и разработок. Его работа оказала значительное влияние на эту область и продолжает оставаться актуальной и сегодня.

Алгоритм работает следующим образом:

1. Строим минимальное остовное дерево (MST).
2. Ищем совершенное паросочетание минимального веса на вершинах дерева нечётной степени.
3. Создаем мультиграф путём добавления рёбер совершенного паросочетания к минимальному остовному дереву.
4. Ищем эйлеров цикл на мультиграфе.
5. Преобразуем эйлеров цикл в гамильтонов путем удаления повторяющихся вершин.

Алгоритм Кристофида использует свойства MST и Эйлера для построения хорошего решения задачи RPP путём создания сбалансированного мультиграфа, который посещает все рёбра, и он гарантирует

получение решения с аппроксимационным коэффициентом 3/2. Этот алгоритм относительно прост в реализации, хорошо работает на многих графах и считается одним из лучших аппроксимационных алгоритмов для задачи RPP.

Техническая часть

Необходимые определения и теоремы

Представим некоторые определения и теоремы для понимания RPP:

Граф, как математический объект, есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер

Гамильтонов цикл - это цикл, который посещает каждую вершину в графе ровно один раз.

Индуцированными называют подграфы, в которых все вершины включает в себя все ребра исходного графа, инцидентные этой вершине.

Мультиграф - граф, в котором разрешается присутствие кратных рёбер.

Аппроксимационный алгоритм - алгоритм, использующийся для поиска приближённого решения оптимизационной задачи.

Эйлеровость

Граф *эйлеров*, если есть цикл, посещающий все ребра по одному разу (быть может посещая некоторые вершины более одного раза или же не посещая их).

Граф *полуэйлеров*, если есть путь, посещающий все ребра по одному разу (быть может посещая некоторые вершины более одного раза или же не посещая их).

Эйлеров цикл - цикл, который посещает каждое ребро в графе ровно один раз.

Теорема (б/д). Связный граф эйлеров тогда и только тогда, когда в нем все вершины имеют четную степень.

Теорема (б/д). Связный граф полуэйлеров, если все вершины кроме двух (или кроме нуля) имеют нечетную степень.

Algorithm. Поиск эйлерова цикла.

Допустим, ответ существует (проверяем критерий).Будем делать DFS по ребрам, то есть запускать обычный DFS, только посещать не вершины, а ребра в массиве *used*. Тогда алгоритм имеет такой вид:

- Переберем все ребра из вершины v, в ходе перебора «удаляем» ребро из графа и рекурсивно

вызываем от второго конца ребра.

- На моменте выхода из рекурсии пишем вершину в итоговый цикл.

Утверждение. Алгоритм выше корректен. Все доказательства в курсе дискретного анализа.

Лемма о безопасном ребре

Остовом графа $G = (V, E)$ будем называть граф $H = (V, E')$, где $E' \subseteq E$.

Остовным деревом графа $G = (V, E)$ будем называть остов, образующий дерево.

Минимальным остовным деревом графа $G = (V, E, w)$ будем называть такое остовное дерево $H = (V, E', w)$, что $\sum\limits_{e \in E'} w(e) \to \min$

$\langle S, T\rangle$ — *разрез*, если $S \cup T = V, S \cap T = \varnothing$

(u, v) пересекает разрез $\langle S, T\rangle$, если u и v в разных частях разреза.

Пусть $G' = (V, E')$ — подграф некоторого минимального остовного дерева G. Ребро $(u, v) \notin G'$ называется *безопасным*, если при добавлении его в G', $G' \cup \{(u, v)\}$ также является подграфом некоторого минимального остовного дерева графа G.

Лемма. Рассмотрим связный неориентированный взвешенный граф $G = (V, E)$ с весовой функцией $w : E \to \mathbb{R}$. Пусть $G' = (V, E')$ — подграф некоторого минимального остовного дерева G, $\langle S, T\rangle$ — разрез G, такой, что ни одно ребро из E' не пересекает разрез, а (u, v) — ребро минимального веса среди всех ребер, пересекающих разрез $\langle S, T\rangle$. Тогда ребро $e = (u, v)$ является безопасным для G'.

Доказательство: Достроим E' до некоторого минимального остовного дерева, обозначим его T_{min}. Если ребро $e \in T_{min}$, то лемма доказана, поэтому рассмотрим случай, когда ребро $e \notin T_{min}$. Рассмотрим путь в T_{min} от вершины u до вершины v. Так как эти вершины принадлежат разным долям разреза, то хотя бы одно ребро пути пересекает разрез, назовем его e'. По условию леммы $w(e) \leqslant w(e')$. Заменим ребро e' в T_{min} на ребро e. Полученное дерево также является минимальным остовным деревом графа G, поскольку все вершины G по-прежнему связаны и вес дерева не увеличился. Следовательно $E' \cup \{e\}$ можно дополнить до минимального остовного дерева в графе G, то есть ребро e — безопасное. ■

Паросочетания

Паросочетанием в неориентированном графе $G = (V, E)$ называют множество ребер $M \subseteq E$ такое, что не найдется двух ребер из M с общей вершиной.

Паросочетание M называется *совершенным или максимальным*, если не существует паросочетания M' такого, что $|M| < |M'|$.

Вершина называется *насыщенной паросочетанием* M, если она является концом какого-то ребра из M.

Увеличивающая цепь относительно паросочетания M — путь $p = (v_1, \dots, v_{2k})$ такой, что $(v_1, v_2) \notin M$, $(v_2, v_3) \in M$, $(v_3, v_2) \notin M$, $\dots$, $(v_{2k-1}, v_{2k}) \notin M$, при этом вершины v_1 и v_{2k} не насыщены M.

Теорема (Бёрдж). Паросочетание M максимально тогда и только тогда, когда относительно M нет увеличивающих цепей.

Доказательство:

$\Longrightarrow$ Докажем методом от противного. Рассмотрим паросочетание M и увеличивающую цепь относительно него. Проведем *чередование* вдоль нее, то есть все ребра из паросочетания на этом пути удалим из него, а ребра из пути, отсутствовавшие в M, добавим. Полученное множество все еще будет паросочетанием, так как крайние вершины пути не были насыщенны M, а остальные вершины как были насыщенными, так ими и остались. Таким образом, построили M' такое, что $|M'| > |M|$.

$\Longleftarrow$ Пусть относительно M нет увеличивающей цепи, а M' — максимальное паросочетание такое, что $|M'| > |M|$. Рассмотрим $Q = M \triangle M'$ — симметрическую разность двух паросочетаний. В Q степень каждой вершины не превосходит двух, так как она могла быть насыщенна не более чем каждым из паросочетаний.

Лемма. Если в графе степень каждой вершины не превосходит двух, то его ребра разбиваются на непересекающиеся пути и циклы.

Доказательство: Изолированные вершины никак не влияют на множество ребер, поэтому удалим из графа, это не изменит структуру множества ребер.

Пусть нашлась вершина v_1 степени один, тогда рассмотрим смежное ей ребро $e = (v_1, v_2)$. Так как степень v_2 не превосходит двух, после удаления e степень вершины равна либо нулю, либо единице. В первом случае получаем, что удалено ребро, не пересекающееся по вершинам больше ни с чем. Иначе степень v_2 равна единице, тогда применим к ней ту же операцию отрезания ребра. По индукции за конечное число шагов придем к вершине v_k, степень которой станет нулевой после удаления ребра. Тогда построили путь $p = (v_1, \dots, v_k)$ такой, что он не пересекается с другими ребрами по вершинам, а значит можем удалить из графа его вершины.

После конечного числа итераций алгоритма выше получим, что в графе могли остаться только степени вершины два. Рассмотрим какую-нибудь из них и запустим схожую процедуру. Тогда рано или поздно мы придем в ту же вершину, причем не могли пойти никуда иначе, так как степени вершин будут равны единице на каждой итерации. За конечное число шагов вернемся в ту же вершину, иначе мы бы пришли в вершину, изначальная степень которой равна единице, чего быть не может. Значит нашли изолированный цикл, который тоже можно удалить.

Данный алгоритм доказывает требуемое.■

Значит Q является объединением непересекающихся циклов и путей.

1. Рассмотрим циклы. Пусть есть цикл нечетной длины $C = (v_1, \dots, v_{2k}, v_1)$. В данном цикле не могут идти два ребра подряд из одного паросочетания, а значит есть вершина степени два, у которой оба ребра из одного и того же паросочетания. Значит есть только циклы четной длины, причем в них поровну ребер из M и M'.
2. Рассмотрим пути. Пусть есть путь нечетной длины $p = (v_1, \dots, v_{2k})$. Рассмотрим два случая:
   - $(v_1, v_2) \in M$, тогда относительно M' есть увеличивающая цепь, что по доказанной необходимости влечет немаксимальность M'.
   - $(v_1, v_2) \in M'$, тогда относительно M есть увеличивающая цепь, но M определялось как паросочетание без них.

   Откуда все пути четной длины и в них поровну ребер из M и M'.

Получаем, что $|Q \cap M| = |Q \cap M'|$.

Очевидно, что $M = (M \cap Q) \sqcup (M \cap Q^C)$, при этом нетрудно показать, что $M \cap Q^C = M \cap M'$, то есть $M = (M \cap Q) \sqcup (M \cap M')$. Аналогично $M' = (M' \cap Q) \sqcup (M \cap M')$. Откуда

$$|M| = |M \cap Q| + |M \cap M'|$$
$$|M'| = |M' \cap Q| + |M \cap M'|$$
$$|M \cap Q| = |M' \cap Q|$$

Получаем, что $|M'| = |M|$.

■

Подробное описание алгоритма

Пусть $G = (V, w)$ - полный граф на множестве вершин V, и весовая функция w присваивает неотрицательный вещественный вес каждому ребру G. Согласно неравенству треугольника, для каждых трех вершин u, v и x должно быть так, что $w(uv) + w(vx) \geq w(ux)$.

1. Создать минимальное остовное дерево T из G.
2. Пусть O - множество вершин с нечетной степенью в T. Согласно лемме о рукопожатиях, O имеет четное число вершин.
3. Найти идеальное соответствие с минимальным весом M в индуцированном подграфе, заданном вершинами из O.

4. Объединить ребра M и T, чтобы сформировать связный мультиграф H, в котором каждая вершина имеет четную степень.

5. Найти эйлеров цикл в H.

6. Преобразовать цикл, найденный на предыдущем шаге, в гамильтонов, пропуская повторяющиеся вершины.

Шаги 5 и 6 не обязательно дают только один результат. Таким образом, эвристика может дать несколько различных путей.

Теорема. Алгоритм Кристофидеса аппроксимирует задачу RPP с точностью 3/2.

Доказательство: Пусть T - минимальное остовное дерево входного графа G и пусть $C*$ - оптимальное решение RPP на G, тогда стоимость T не превосходит $\frac{2}{3}$ стоимости $C*$. Это связано с тем, что является неоптимальным решением, а вес рёбер в не превышает $\frac{2}{3}$ веса рёбер в $C*$.

Пусть M - совершенное паросочетание вершин нечётной степени в T с минимальным весом и пусть C - мультиграф, образованный добавлением рёбер в M к T, тогда стоимость M не превосходит $\frac{1}{2}$ стоимости C*.

Пусть E - эйлерова цепь C и C' - гамильтонов цикл, полученный путём удаления повторяющихся вершин из E, тогда стоимость C' не превосходит стоимости $C*$.

Суммируя вышесказанное, мы получаем: $C' \leq C \leq M + T \leq \frac{1}{2}C* + \frac{2}{3}C* = \frac{5}{6}C*$

Таким образом, стоимость C' не превосходит $\frac{5}{6}$ стоимости оптимального решения $C*$, это гарантирует, что алгоритм имеет аппроксимационную точность $\frac{3}{2}$.

■

Основная часть

Свойства алгоритма

Покажем, что алгоритм Кристофидеса для решения задачи RPP принадлежит классу NP.

1. Чтобы доказать, что алгоритм Кристофида решает задачу RPP за полиномиальное время, достаточно показать, что каждый шаг алгоритма может быть сделан за полиномиальное время. Алгоритм состоит из нескольких шагов, включая построение минимального остовного дерева, нахождение минимального веса совершенного паросочетания, создание мультиграфа, нахождение эйлерового цикла и преобразование его в гамильтонов. Все эти шаги можно выполнить за полиномиальное время, используя такие алгоритмы, как алгоритмы Крускала или Прима для минимального остовного дерева, алгоритм Блоссома (сжатия цветков) для совершенного паросочетания и алгоритм Иеролцера (Hierholzer's Algorithm) для поиска эйлерового цикла.

2. Чтобы показать, что задача RPP относится к классу NP, мы должны продемонстрировать, что существует алгоритм полиномиального времени для проверки решения. Задача RPP - найти гамильтонов цикл, который посещает каждое ребро в графе ровно один раз и имеет минимальный общий вес. Алгоритм проверки можно выполнить, проверив, что цикл решения посещает каждое ребро ровно один раз и что общий вес является минимальным.Это можно сделать за полиномиальное время, посетив каждое ребро цикла и подсчитав количество посещений, а также убедившись, что оно было посещено один раз.

Поскольку алгоритм Кристофидеса может решить задачу RPP за полиномиальное время, а задача принадлежит NP (так как решение может быть проверено за полиномиальное время), выходит, что алгоритм Кристофидеса для решения задачи RPP принадлежит классу NP.

Имплементация алгоритма

Имплементируем описанный нами алгоритм на языке Python:

Код в виде файла можно найти в репозитории GitHub.

```
import math
import networkx as nx

def christofides_rpp(G):
    # Create a minimum spanning tree of the graph
    MST = nx.minimum_spanning_tree(G)

    # Identify the set of odd-degree vertices in the MST
    odd_vertices = [v for v in MST.nodes() if MST.degree(v) % 2 == 1]

    # Create a minimum weight perfect matching on the set of odd-degree vertices
    matching = nx.bipartite.minimum_weight_full_matching(G, odd_vertices)

    # Combine the MST and the minimum weight perfect matching to form a multi-graph
    multi_graph = nx.MultiGraph(MST)
    for u, v, weight in matching:
        multi_graph.add_edge(u, v, weight=weight)

    # Find an Eulerian tour of the multi-graph
    tour = nx.eulerian_circuit(multi_graph)

    # Convert the Eulerian tour into a Hamiltonian cycle by removing any repeated vertices
```

```
    hamiltonian_cycle = []
    for u, v in tour:
        if u not in hamiltonian_cycle:
            hamiltonian_cycle.append(u)
        if v not in hamiltonian_cycle:
            hamiltonian_cycle.append(v)

    return hamiltonian_cycle
```

Эта функция принимает в качестве входного граф G, представленный в качестве объекта графа networkx, и возвращает список, представляющий аппроксимационное решение RPP в виде гамильтонова цикла. Точность решения - 3/2.

Тесты

Представим несколько тестов для задачи RPP:

Код в виде файла можно найти в репозитории GitHub.

```
import math
import networkx as nx

def test_rpp():
    # Test 1: Trivial case with one vertex and no edges
    G = nx.Graph()
    G.add_node(1)
    solution = christofides_rpp(G)
    assert solution == [1], f"Expected [1] but got {solution}"

    # Test 2: Trivial case with two vertices and one edge
    G = nx.Graph()
    G.add_edge(1, 2, weight=1)
    solution = christofides_rpp(G)
    assert solution == [1, 2, 1], f"Expected [1, 2, 1] but got {solution}"

    # Test 3: Simple case with three vertices and three edges
    G = nx.Graph()
    G.add_edge(1, 2, weight=1)
    G.add_edge(2, 3, weight=2)
    G.add_edge(3, 1, weight=3)
```

```
    solution = christofides_rpp(G)
    assert solution == [1, 2, 3, 1], f"Expected [1, 2, 3, 1] but got {solution}"

    # Test 4: Complex case with four vertices and six edges
    G = nx.Graph()
    G.add_edge(1, 2, weight=1)
    G.add_edge(2, 3, weight=2)
    G.add_edge(3, 4, weight=3)
    G.add_edge(4, 1, weight=4)
    G.add_edge(1, 3, weight=5)
    G.add_edge(2, 4, weight=6)
    solution = christofides_rpp(G)
    assert solution == [1, 2, 3, 4, 1], f"Expected [1, 2, 3, 4, 1] but got {solution}"

test_rpp()
```

Эти тесты охватывают несколько различных случаев, включая тривиальный случай с одной вершиной и без рёбер, тривиальный случай с двумя вершинами и одним ребром, простой случай с тремя вершинами и тремя рёбрами и более сложный случай с четырьмя вершинами и шестью рёбрами.

Альтернативные алгоритмы

Одной из основных альтернатив алгоритму Кристофида является алгоритм Лина - Кернигана. В алгоритме предполагается, что некоторое начальное разбиение графа уже существует, затем имеющееся приближение улучшается в течение некоторого количества итераций. Применяемый способ улучшения состоит в обмене вершинами между подмножествами имеющегося разбиения графа. Для формирования требуемого количества частей графа может быть использована рекурсивная процедура деления пополам. Этот алгоритм имеет очень долгое время вычисления, может занять даже несколько дней, чтобы решить сложный случай задачи RPP.

Другой альтернативой алгоритма Кристофидеса является алгоритм ветвления и разреза (branch and cut), который способен решить сложные случаи задачи RPP очень быстро. Этот алгоритм позволяет легко включить другие ограничения. Однако это требует много памяти и вычислительной мощности.

Другим подходом является использование генетических алгоритмов и имитация отжига, эти методы были использованы для решения проблемы RPP с хорошими результатами, но они не так эффективны, как другие ранее упомянутые.

В итоге, алгоритм Кристофида является относительно простым алгоритмом для реализации и хорошо работает на многих графах, в то время как другие алгоритмы, такие как алгоритм Лина-Кернигана, алгоритм ветвления и разреза или метаэвристические алгоритмы, такие как генетиче-

ские алгоритмы или имитация отжига, были показаны более эффективными в некоторых случаях, но они требуют больше вычислительных ресурсов.

Применение задачи на практике

RPP имеет широкий спектр практических применений в таких областях, как транспорт, логистика и проектирование сетей.

Вот несколько примеров:

1. Почтовая доставка: в контексте почтовой доставки, RPP может быть использован для оптимизации маршрута почтальона, так что все дома в данном районе посещаются с минимальным общим расстоянием.
2. Общественный транспорт: RPP можно использовать для нахождения оптимальных маршрутов для автобусов и поездов, чтобы обслуживать все остановки или станции в данном районе с минимальным общим расстоянием.
3. Электроэнергетика: RPP может использоваться для определения наиболее эффективных маршрутов для электромобилей для обслуживания всех трансформаторов и подстанций в данном районе с минимальным общим расстоянием.
4. Задача коммивояжера: RPP можно использовать для оптимизации маршрутов продавцов, техников или других полевых работников, чтобы посетить всех клиентов, клиентов или мест с минимальным общим расстоянием.
5. Оптимизация логистики и цепочки поставок, а именно минимизация расстояния и расхода топлива у грузовых автомобилей для обслуживания всех клиентов/складов в определенном районе.

В целом, RPP можно применить к любой проблеме, требующей посещения всех точек в сети с минимальным общим расстоянием.

Заключение

Литература

1. An algorithm for RPP, N. Christofides, V. Campos, A. Corberan, E. Mota
2. Задачи китайского и деревенского почтальонов, Рене Андреасович ван Беверн
3. On the Generalized DRPP, M. Drexl

4. Chinese postman problem, Wikipedia

5. Алгоритм Кристофидеса, Википедия