Сложность вычислений

Метрическая задача коммивояжера

Андрей Саутин
Группа 396
ФИВТ, МФТИ

2015

1 Введение

Задача коммивояжера (Travelling Salesman Problem, TSP), известная математическому миру уже более двухсот лет, изначально возникла как чисто развлекательная. Вскоре же она превратилась в серьезную научную проблему [9] и нашла активное применение во всевозможных областях: планирование, логистика, производство микрочипов, секвенирование ДНК, астрономия и др. [8]

Со времен рождения задачи и до наших дней разными учеными было предприняты попытки в ее решении. Среди тех, кто работал над ней, встречаются и такие известные математики, как Леонард Эйлер (1707-1783), Уильям Роуан Гамильтон (1805-1865), Джордж Бернард Данциг (1914-2005), Ричард Беллман (1920-1984). Часть результатов их трудов состоит в великом множестве алгоритмов, решающих различные вариации задачи коммивояжера с той или иной точностью. [9] Тем не менее, несмотря на простоту формулировки, полиномиальное решение для общего случая задачи TSP до сих пор найдено не было. В 1972 году Ричард Карп в своей работе "Reducibility Among Combinatorial Problems"показал, что частный случай задачи коммивояжера — задача поиска гамильтонова цикла в неориентированном графе — является NP-полной. [5]

В этой работе сформулированы и доказаны некоторые утверждения, касающиеся задачи коммивояжера, а также описаны и реализованы на языке C++11 две эвристики для нахождения приближенного решения метрической задачи TSP.

2 Вариации задачи коммивояжера

Общая задача. Задан граф $G = (V, E, w)$, где V — множество вершин, E — множество ребер, $w : E \to \mathbb{R}_+$ — весовая функция на ребрах. Необходимо найти цикл $u_1, ..., u_n$, где $n = |V|$, проходящий по каждой вершине из V ровно один раз, т.е. $\forall v \in V \; \exists! i \in \{1, .., n\} : u_i = v$.
Эту же задачу можно сформулировать следующим образом: найти в графе $G = (V, E, w)$ гамильтонов цикл минимального веса.

Метрическая задача. Частный случай общей задачи. Граф G — полный, а весовая функция должна удовлетворять неравенству треугольника, т.е. $\forall x, y, z \in V : w(x, z) \leq w(x, y) + w(y, z)$.

Евклидова (планарная) задача. Частный случай метрической задачи. Вершины графа G — точки на плоскости, а весовая функция на ребрах определяется как евклидово расстояние между ними. [10]

3 Сложность задачи коммивояжера

Теорема 1 (б/д)**.** *Язык HAMCYCLE графов, содержащих гамильтонов цикл, является NP-полным. [1]*

Теорема 2. *Если $P \neq NP$, то у общей задачи коммивояжера не существует полиномиальных константных алгоритмов приближения.*

Доказательство. Предположим противное: пусть существует α-приближающий алгоритм решения общей задачи коммивояжера. Рассмотрим произвольный невзвешенный граф G на n вершинах, присвоим всем его ребрам вес 1. Получим граф G' из G следующим образом: между любыми двумя вершинами в G, не соединенными ребром, добавим ребро веса $\alpha n + 1$.
Заметим, что если в исходном графе G был гамильтонов цикл, то в графе G' есть гамильтонов цикл веса n. Если же в G не было гамильтонова цикла, то в G' самый "легкий"гамильтонов цикл имеет вес хотя бы $(\alpha n+1)+n-1 > \alpha n$. С помощью α-приближающего алгоритма можем понять, превосходит ли стоимость оптимального цикла в G' значение n. Но тогда также становится возможным ответить на вопрос о существовании гамильтонова цикла в G. В силу произвольности G получаем, что существует полиномиальный алгоритм, разрешающий язык HAMCYCLE. Тогда HAMCYCLE$\in P$. В силу NP-полноты языка HAMCYCLE: $P = NP$, что противоречит условию.
Итак, у общей задачи коммивояжера не существует полиномиальных константных алгоритмов приближения. [11] □

Теорема 2 показывает, что общую задачу коммивояжера не получится хорошо аппроксимировать. Тем не менее для некоторых задач с дополнительными ограничениями на весовую функцию существуют полиномиальные константные алгоритмы приближения. В последующих разделах описаны два наиболее известных константных алгоритма аппроксимации для метрической задачи TSP.

4 2-приближение метрической задачи коммивояжера

4.1 Описание алгоритма

Алгоритм 1. *Пусть задан полный граф $G = (V, E, w)$, причем для весовой функции w выполняется неравенство треугольника.*

1. *Найдем минимальное остовное дерево T (дерево на всех вершинах графа G, суммарный вес ребер которого минимален среди всех таких деревьев) в графе G.*
2. *Обойдем дерево T в глубину и запишем вершины без повторов в порядке их посещения. Более строго: для каждой вершины найдем время первого входа в нее, затем отсортируем вершины по этому параметру. Получим некоторый путь P. Построим гамильтонов цикл H, который обходит вершины графа G в порядке, задаваемом путем P (пропуская повторяющиеся вершины).*

[11]

Лемма 1. *Пусть P — произвольный цикл, обходящий все вершины графа G, а H — гамильтонов цикл в G, обходящий вершины графа в том же порядке, что и P (пропуская повторяющиеся вершины). Тогда $w(H) \leq w(P)$. Здесь и далее будет использоваться обозначение $w(A) = \sum\limits_{e \in A} w(e)$, где $A \subset E$.*

Доказательство. В цикле $P = (v_1, .., v_N)$ оставим только первое посещение каждой вершины: пусть $P = (v_1, .., v_{k-i_1-1}, .., v_{k-1}, v_k, v_{k+1}, .., v_{k+i_2+1}, .., v_N)$ и вершины $v_{k-i_1}, .., v_{k+i_2}$ уже были посещены ранее. Тогда, удаляя их, мы добавляем ребро $e^* = (v_{k-i_1-1}, v_{k+i_2+1})$, причем $w(e^*) \leq w((v_{k-i_1}, v_{k-i_1+1})) + ... + w((v_{k+i_2-1}, v_{k+i_2}))$ по неравенству треугольника. После удаления всех повторяющихся вершин получим гамильтонов цикл H, причем $w(H) \leq w(P)$ из неравенства треугольника. □

Лемма 2. *Пусть H — гамильтонов цикл в графе G, а T — минимальное остовное дерево в G. Тогда $w(T) \leq w(H)$.*

Доказательство. При удалении произвольного ребра из цикла H получаем остовное дерево T' и $w(T') \leq w(H)$. При этом T — минимальное остовное дерево, следовательно, $w(T) \leq w(T')$. Итого: $w(T) \leq w(T') \leq w(H)$. □

Утверждение 1. *Алгоритм 1 является 2-приближающим алгоритмом с полиномиальным временем работы для метрической задачи TSP.*

Доказательство. Построение минимального остовного дерева для полного графа выполняется за время $O(|V|^2)$ с помощью алгоритма Прима (без использования кучи). Обход дерева в глубину выполняется за $O(|E|)$. В итоге получаем, что tspApproximationMST является полиномиальным алгоритмом, т.к. выполняется за время $O(|V|^2)$.
Обозначим за H^* гамильтонов цикл минимального веса.
Пусть W — обход дерева T в глубину. Тогда $w(W) = 2 \cdot w(T)$, т.к. по каждому ребру мы проходим дважды. При этом W — это обход всех вершин графа, следовательно, по лемме 1 $w(H^*) \leq w(W)$. Кроме того, по лемме 2: $w(T) \leq w(H)$. Итого: $w(H^*) \leq w(W) = 2 \cdot w(T) \leq 2 \cdot w(H) \Rightarrow w(H^*) \leq 2 \cdot w(H)$.
Таким образом, алгоритм 1 является 2-приближающим алгоритмом с полиномиальным временем работы.[11] □

4.2 Реализация алгоритма на C++11

tspApproximationMST

```
1 Tour tspApproximationMST(const CompleteGraph& graph) {
2     Graph mst = minimalSpanningTree(graph);
3     Tour cycle = createHamiltonianCycle(mst);
4     return cycle;
5 }
```

Алгоритм tspApproximationMST реализуется строго по описанным выше пунктам. Сначала в строке 2 строится минимальное остовное дерево с помощью алгоритма Прима без использования кучи (время: $O(|E|)$, если граф

полный). Затем в строке 3 выполняется обход дерева и удаление повторяющихся вершин (время: $O(|E|)$). Итак, получаем алгоритм 2-приближения метрической задачи TSP, работающий полиномиальное, а именно $O(|V|^2)$ время.

5 $\frac{3}{2}$-приближение метрической задачи коммивояжера

5.1 Описание алгоритма

Алгоритм 2 (Кристофидес)**.** *Пусть задан полный граф $G = (V, E, w)$, причем для весовой функции w выполняется неравенство треугольника.*

1. *Найдем минимальное остовное дерево T (дерево на всех вершинах графа G, суммарный вес ребер которого минимален среди всех таких деревьев) в графе G.*
2. *Построим полный граф M' на вершинах нечетной степени в дереве T.*
3. *Найдем совершенное паросочетание M минимального веса в M', т.е. найдем такой набор ребер наименьшего суммарного веса, что для любой вершины найдется ровно одно инцидентное ей ребро, лежащее в наборе.*
4. *Построим мультиграф G' на вершинах графа G, объединив множество ребер дерева T и паросочетания M.*
5. *Построим эйлеров цикл C в G'.*
6. *Построим гамильтонов цикл H, который обходит вершины графа G в порядке, задаваемом путем C, т.е., по сути, удалим из C повторяющиеся вершины.*

[11]

Лемма 3. *Пусть H^* — минимальный гамильтонов цикл в графе $G = (V, E)$, а M — совершенное паросочетание минимального веса в G. Тогда $w(H^*) \geq 2w(M)$.*

Доказательство. Пусть $H^* = (v_1, ..., v_n)$, где $v_i \in V, \forall i \in \{1, ..., n\}$. Тогда $M_1 = (v_1, v_2), ..., (v_{n-1}, v_n)$ и $M_2 = (v_2, v_3), ..., (v_{n-2}, v_{n-1}), (v_n, v_1)$ — два паросочетания в графе G, причем $w(H^*) = w(M_1) + w(M_2)$, т.к. ребра паросочетаний M_1, M_2 образуют разбиение ребер цикла H^*. При этом вес каждого из них не меньше веса минимального паросочетания, т.е. $w(M_1) + w(M_2) \geq w(M) + w(M)$. В итоге получаем: $w(H^*) \geq 2w(M)$. □

Утверждение 2. *Алгоритм Кристофидеса корректен и является $\frac{3}{2}$-приближающим алгоритмом с полиномиальным временем работы для метрической задачи TSP.*

Доказательство.
1. Полиномиальность. Построение минимального остовного дерева для полного графа выполняется за время $O(|V|^2)$ с помощью алгоритма Прима (без использования кучи). Выделение вершин нечетной степени и построение индуцированного полного графа на них — за время $O(|E|)$. Поиск совершенного паросочетания минимального веса реализуется, например, алгоритмом Blossom V за $O(|V|^3|E|)$. Построение эйлерова графа G', включающее объединение ребер дерева и паросочетания, выполняется за $O(|E|)$. Поиск эйлерова цикла C в графе G' реализуется с помощью модифицированного поиска в глубину, занимает время $O(|E|)$. Наконец, удаление из C повторяющихся вершин занимает $O(|V|)$ времени. Итого, алгоритм Кристофидеса имеет временную сложность $O(|V|^3|E|) = O(|V|^5)$, т.е. алгоритм Кристофидеса является полиномиальным алгоритмом.
2. Корректность. Для доказательства корректности алгоритма необходимо показать, что для произвольного полного взвешенного графа можно проделать описанные в алгоритме действия и получить гамильтонов цикл.

Почему всегда можно построить совершенное паросочетание M на графе M' (см. п.2 алгоритма)? Заметим, что совершенное паросочетание в полном графе можно построить тогда и только тогда, когда количество вершин в нем четно. В любом графе сумма степеней вершин равна удвоенному числу ребер, т.е. эта сумма является четным числом. Но тогда получается, что в любом графе количество вершин нечетной степени четно. Таким образом, в M' существует совершенное паросочетание.

Почему мультиграф G' — эйлеров? Пусть $\{v_1, ..., v_n\}$ — вершины нечетной степени дерева T. Паросочетание M построено только на этих вершинах, причем (по определению совершенного паросочетания) $deg_M(v_i) = 1$, $\forall i \in \{1, ..., n\}$, где $deg_M(v)$ — степень вершины v в графе M. Но тогда, добавляя паросочетания M в дерево T, получаем мультиграф G', в котором степени всех вершин четные. Это и означает, что G' эйлеров.

3. Показатель аппроксимации. По лемме 1: $w(H) \leq w(C)$. При этом эйлеров цикл проходит по всем ребрам мультиграфа G', а множество его ребер получено дизъюнктным объединением множества ребер T и множества ребер M: $w(C) = w(G') = w(T) + w(M)$.
Обозначим за H^* гамильтонов цикл минимального веса в G, а за H^*_M гамильтонов цикл минимального веса в M'. При этом по лемме 3: $w(M) \leq \frac{1}{2}w(H^*_M)$. В свою очередь $w(H^*_M) \leq w(H^*)$, т.к. при удалении из H^* вершин, не лежащих в M', получаем некий гамильтонов цикл на вершинах M', а его вес не меньше веса H^*_M по неравенству треугольника.
В то же время по лемме 2: $w(T) \leq w(H^*)$.

Собирая все неравенства вместе, получаем: $w(H) \leq w(C) = w(G') = w(T) + w(M) \leq w(H^*) + \frac{1}{2}w(H^*_M) \leq w(H^*) + \frac{1}{2}w(H^*) = \frac{3}{2}w(H^*_M)$.
Таким образом, алгоритм Кристофидеса является $\frac{3}{2}$-приближающим для метрической задачи коммивояжера. □

5.2 Реализация алгоритма на C++11

tspApproximationChristofides

```
1  Tour tspApproximationChristofides(const CompleteGraph& graph) {
2      Graph mst = minimalSpanningTree(graph);
3      std::vector<int> odd_degree_nodes;
4      for (size_t node = 0; node < mst.size(); ++node) {
5        if (nodeDegree(mst, node) % 2 == 1) {
6          odd_degree_nodes.push_back(node);
7        }
8      }
9
10     InduceResult induce_result = induce(graph,
   ↪  odd_degree_nodes);
11     Matching matching =
   ↪  perfectMinWeightMatching(induce_result.graph);
12     for (Edge& edge : matching) {
13       edge.from = induce_result.ind_to_init[edge.from];
14       edge.to   = induce_result.ind_to_init[edge.to  ];
15     }
16     std::vector<Edge> mst_edges = mst.getEdges();
17     std::vector<Edge> edges(matching.size() + mst_edges.size());
18     auto it = std::copy(mst_edges.begin(), mst_edges.end(),
   ↪  edges.begin());
19     std::copy(matching.begin(), matching.end(), it);
20
21     std::vector<int> eulerian_cycle =
   ↪  eulerianCycle(Graph(graph.size(), edges));
22     Tour cycle = createCycleOnTour(graph.size(),
   ↪  eulerian_cycle);
23     return cycle;
24 }
```

Алгоритм реализуется строго по описанным выше пунктам. Сначала в строке 2 строится минимальное остовное дерево T с помощью алгоритма Прима без использования кучи (время: $O(|E|)$, если граф полный).
Затем в строках 3-8 из дерева T выделяются все вершины нечетной степени простым проходом по всем вершинам ($O(|V|)$).
В строке 10 исходный граф G индуцируется на множество вершин дерева нечетной степени, давая в результате полный граф M'. При этом перебираются все ребра исходного графа и добавляются только те, оба конца которых лежат в множестве вершин индуцированного графа. Также строится взаимно-однозначное соответствие между номерами вершин в индуцированном подграфе и их же номерами в исходном. Все операции на этом этапе суммарно занимают $O(|E|)$ времени.
После этого в строке 11 в графе M' ищется совершенное паросочетание M минимального веса. Для решения этой подзадачи был выбран жадный алгоритм, который дает приближенное значение оптимального совершенного паросочетания. В [7] доказано, что отношение найденного жадным

алгоритмом паросочетания к оптимальному не превышает $\frac{4}{3}|V|^{lg(\frac{3}{2})} - 1 \simeq \frac{4}{3}|V|^{0.585} - 1$, причем эта оценка достигается. Таким образом, жадный алгоритм дает достаточно плохое приближение оптимального совершенного паросочетания. Изначально было задумано реализовать один из точных алгоритмов Blossom IV или Blossom V, описанные в [4] и [6] соответственно, однако, ввиду масштабности и сложности этих решений, в итоге был выбран жадный алгоритм. Последний устроен следующим образом: на каждой из $\frac{|V|}{2}$ итераций находим две вершины, еще не лежащие в паросочетании, ребро между которыми имеет минимальный вес; добавляем эти вершины и ребро, их связывающее, в паросочетание и переходим к следующей итерации. В целях эффективности реализации предлагается не искать минимальное расстояние на каждой итерации, а отсортировать массив всех ребер и добавлять их по очереди, соединяя вершины, еще не лежащие в паросочетании. Время работы жадного алгоритма: $O(|V|^2 log|V|)$. [7, 2]
В строках 16-19 строится множество ребер мультиграфа G' обычным копированием ребер дерева и паросочетания (время: $O(|E|)$).
Затем, в строке 21 в мультиграфе G' выделяется эйлеров цикл. Для этого запускается модифицированный обход в глубину, помечающий в процессе обхода посещенные ребра, а не вершины. Он проходит по каждому ребру единожды, поэтому этот пункт алгоритма выполняется за $O(|E|)$.
Наконец, в строке 22 по эйлерову циклу строится гамильтонов цикл путем удаления повторяющихся вершин (время: $O(|V|)$).
Итак, получаем алгоритм приближения метрической задачи TSP, работающий полиномиальное, а именно — $O(|V|^2 log|V|)$, время.

6 Тестирование

Оба алгоритма были протестированы на одном и том же наборе тестов. Тесты можно условно разделить на две группы:

1. Непланарная задача TSP. Несколько тестов, входящих в эту группу, представляют собой простые примеры метрической непланарной задачи коммивояжера, для которых можно легко вычислить вес оптимального гамильтонова цикла.

2. Планарная задача TSP. Более обширный набор тестов, взятых в [3]; примеры планарной задачи коммивояжера, в которой точками являются города некоего государства.

При этом для каждого алгоритма на каждом тесте проверялась точность выдаваемого ответа. Реализации обоих решений прошли все тесты, причем оба алгоритма подтвердили свой теоретический показатель аппроксимации:

- алгоритм tspApproximationMST на наборе тестов выдавал ответы, отличающиеся от оптимальных максимум в 1.39;

- алгоритм tspApproximationChristofides (несмотря на реализацию поиска паросочетания через жадный алгоритм) на наборе тестов выдавал ответы, отличающиеся от оптимальных максимум в 1.27.

Кроме того, алгоритм tspApproximationChristofides только на одном тесте (sahara) сработал хуже, чем алгоритм tspApproximationMST. В остальных случаях результаты работы алгоритмов либо совпадали, либо реализация алгоритма Кристофидеса выдавала значения, более близкие к оптимальным.

Список литературы

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*.

[2] Sgall Jiri Chatterjee, Krishnendu. Mathematical foundations of computer science 2013.

[3] William Cook. National traveling salesman problems.

[4] William Cook and Andre Rohe. Computing minimum-weight perfect matchings.

[5] Richard M. Karp. *Reducibility Among Combinatorial Problems*.

[6] Vladimir Kolmogorov. Blossom v: a new implementation of a minimum cost perfect matching algorithm.

[7] Edward M. Reingold and Robert E. Tarjan. On a greedy heuristic for complete matching.

[8] Wikipedia. Travelling salesman problem.

[9] Мудров В.И. *Задача о коммивояжёре*. Знание.

[10] Википедия. Задача коммивояжера.

[11] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, and Клиффорд Штайн. *Алгоритмы. Построение и анализ*. Вильямс.