Московский физико-технический институт (ГУ)
Физтех-школа прикладной математики и информатики

Дерево Штейнера

Алгоритм 2-аппроксимации и его анализ

Андрющенко Соломон
Б05-222

Сложность вычислений, осень 2024

Аннотация

Задача минимального дерева Штейнера является одной из фундаментальных в области комбинаторной оптимизации и часто встречается в практических приложениях, таких как проектирование коммуникационных сетей и схемотехники. В данной работе автор исследует версию задачи дерева Штейнера на графах. Он доказывает **NP**-полноту данной задачи и предлагает алгоритм, который гарантирует приближённое решение с коэффициентом приближения 2.

Содержание

1 Введение

1.1 Постановка задачи

Дан неориентированный связный граф $G = (V, E)$, в нём выделено множество вершин V_0. Также имеются веса на рёбрах $w : E \to \mathbb{R}_+$. Требуется найти дерево T минимального веса, покрывающее все вершины V_0.

1.2 Постановка метрической версии задачи

Дан неориентированный полный граф $G = (V, E)$, в нём выделено множество вершин V_0. Также имеются веса на рёбрах $w : E \to \mathbb{R}_+$ и выполнено неравенство треугольника: $\forall x,\, y,\, z :\ w(x,\, z) \leqslant w(x,\, y) + w(y,\, z)$. Требуется найти дерево T минимального веса, покрывающее все вершины V_0.

2 NP-полнота

Будем доказывать две сводимости: от задачи Chromatic Number к задаче Exact Cover и от задачи Exact Cover к Steiner Tree. **NP**-полнота задачи Chromatic Number предполагается известной. Автор нашёл доказательства в [1] (стр. 15).

Задача Chromatic Number:

> INPUT: Граф G, натуральное число k.
> PROPERTY: Существует фукнция $\phi : \mathbb{N} \to \mathbb{Z}_k$, такая что, если вершины с номерами u и v соединяются ребром, то $\phi(u) \neq \phi(v)$.

Задача Exact Cover:

> INPUT: Семейство $\{S_j\}$ подмножеств $\{1, \ldots, n\}$.
> PROPERTY: Существует подсемейство $\{T_h\} \subseteq \{S_j\}$, такое что все T_h не пересекаются и $\bigcup T_h = \bigcup S_j = \{1, \ldots, n\}$.

Задача Steiner Tree:

> INPUT: Граф $G = (V, E)$, подмножество вершин $V_0 \subseteq V$, функция весов на рёбрах $w : E \to \mathbb{Z}_+ \cup \{0\}$, натуральное число k.
> PROPERTY: G имеет поддерево, веса не более k, содержащее в себе все вершины V_0.

Заметим, что проверить ответ в задаче Steiner Tree можно за полиномиальное время. Действительно, нужно только проверить валидность поддерева и принадлежность всех вершин V_0 этому поддереву, а также что сумма на рёбрах не больше k. Всё это можно сделать за линейное время от размера E. Поэтому эта задача лежит в **NP**.

2.1 Chromatic Number $\leqslant_p$ Exact Cover

Обозначим за $E(v)$ — множество инцидентных рёбер вершине v. Понятно, что вместо множества $M = \{1, \ldots, n\}$ в задаче Exact Cover мы можем брать любое конечное множество, поэтому:

$$\begin{cases} M = V \cup E \cup \{(v, e, i) : v \in V,\, e \in E(v),\, i \in \{1, \ldots, k\}\} \\ S_j = \left[\begin{array}{l} \{v\} \cup \{(v, e, i) : e \in E(v)\},\ v, i \text{ — fixed.} \\ \{e\} \cup \{(v_1, e, i) : i \neq t_1\} \cup \{(v_2, e, j) : j \neq t_2\},\ e, v_1, v_2, t_1, t_2 \text{ — fixed,} \\ \qquad \text{where } v_1,\, v_2 \text{ are incident to } e \text{ and } t_1, t_2 \in \{1, \ldots, k\}, t_1 \neq t_2. \end{array} \right. \end{cases}$$

Пояснение: для каждой вершины мы фиксируем цвет (число $i \in \{1, \ldots, k\}$) с помощью множеств S_j первого типа, а с помощью множеств второго типа мы проверяем, что у соседних вершин не совпадает цвет.

2.2 Exact Cover $\leqslant_p$ Steiner Tree

$$\begin{cases} V = \{n_0\} \cup \{S_j\} \cup \{1, \ldots, n\} \\ V_0 = \{n_0\} \cup \{1, \ldots, n\} \\ E = \{(n_0, S_j)\} \cup \{(S_j, u) : u \in S_j\} \\ w((n_0, S_j)) = |S_j| \\ w((S_j, u)) = 0 \\ k = n \end{cases}$$

Пояснение: мы хотим чтобы семейство $\{S_j\}$ покрывало все вершины $\{1, \ldots, n\}$, поэтому добавляем их всех в терминальные (в V_0). Также добавляем в терминальные фиктивную вершину n_0, которая является как бы связующим звеном между разными множествами S_j. Так как мы хотим брать только непересекающиеся S_j, то ставим $k = n$, и $w((n_0, S_j)) = |S_j|$.

Замечание: если у нас найдётся такое поддерево веса не более $k = n$, то у него вес будет в точности равен n и в $\{T_h\}$ будут входить множества S_j, соответствующие вершинам S_j поддерева.

3 Алгоритм

Автор нашёл алгоритмы сведения к метрической версии задачи и нахождения 2-приближения в [2].

3.1 Сведение к метрической версии

Положим за $b(x, y)$ — вес кратчайшего (по сумме весов на рёбрах) пути от вершины x до вершины y. Заметим, что выполняется неравенство треугольника: $\forall x, y, z : b(x, z) \leqslant b(x, y) + b(y, z)$, так как в правой части неравенства накладывается ограничение на кратчайший путь от x до z — он должен проходить через y. Заметим, что такое преобразование можно выполнить за полиномиальное время от количества вершин воспользовавшись алгоритмом Флойда (точное время работы: $\Theta(|V|^3)$). Теперь докажем, что полученная метрическая версия задачи эквивалентна исходной:

Допустим мы нашли минимальное по весу поддерево T, содержащее V_0, в метрической версии задачи. Заметим, что мы можем в исходной задаче найти поддерево такого же веса или даже меньше: нужно для каждого ребра (x, y) из T брать кратчайший путь из x в y в исходном графе. Причём, заметим, что верно и обратное, а именно: если мы нашли минимальное по весу поддерево T, содержащее V_0, в исходной версии задачи, то если мы возьмём то же поддерево (со всеми его рёбрами) в метрической версии задачи, то суммарный вес дерева может только уменьшиться (так как вес каждого ребра заменён теперь на что-то меньшее, а именно на вес кратчайшего пути между парой вершин).

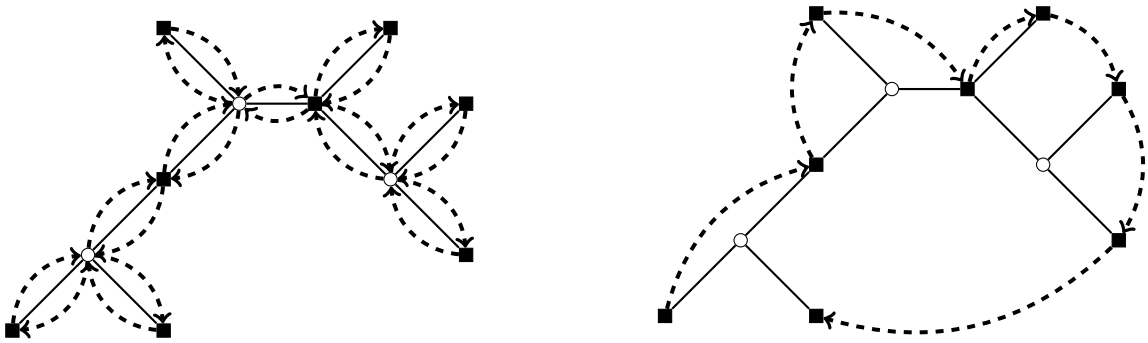
Следовательно, веса минимальных поддеревьев из исходной и из метрической версий задач совпадают. Также, понятно, как имея решение одной задачи сразу получить решение другой (по алгоритму из доказательства выше).

3.2 2-приближение

Приведём полиномиальный алгоритм, который для метрической версии задачи (а значит и для исходной) будет находить подходящее поддерево веса не более удвоенного оптимума (называется 2-аппроксимацией или 2-приближением).

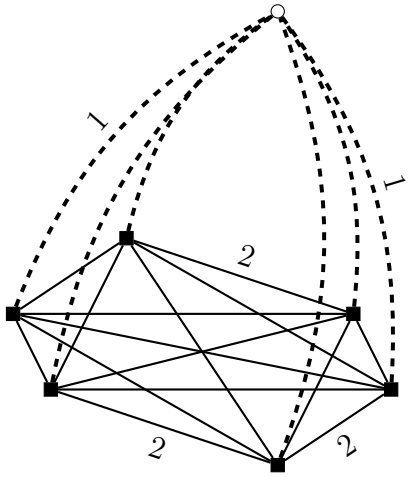
Допустим у нас есть оптимальное поддерево. Давайте раздвоим каждое ребро (создадим на этом месте два противоположных ориентированных ребра) этого поддерева. Следовательно, в этом новом ориентированном графе будет Эйлеров цикл. Далее, возьмём какую-нибудь вершину из V_0 (будем называть такие вершины терминальными) за «первую» и пойдём по этому циклу до следующей не посещённой терминальной вершины, постепенно стягивая рёбра графа (от такой операции суммарный вес полученного дерева не увеличится по неравенству треугольника). Так повторим для «второй» и «третьей» терминальной вершины и так далее, пока не исчерпаем все терминальные вершины. Заметим, что таким образом мы получили новое поддерево только на терминальных вершинах, у которого суммарный вес не более удвоенного веса исходного оптимального поддерева.

Теперь воспользуемся тем, что наше новое поддерево использует только терминальные вершины. Задача сводится к нахождению минимального остова для графа, что можно сделать за $O(|E| \cdot log(|V|))$ по времени с помощью алгоритма Прима, что, конечно, является полиномом от $|V|$.



■ $\in V_0$ ○ $\notin V_0$

Также приведём пример таких графов, где наш алгоритм даёт почти в 2 раза больший ответ, чем оптимальный, тем самым показав, что наша оценка точная. Для этого нужно взять $|V_0| = |V| - 1$ и терминальные вершины образуют клику, где каждое ребро веса 2. Осталась одна нетерминальная вершина, которую соединим со всеми терминальными вершинами, рёбрами веса 1:



■ $\in V_0$ ○ $\notin V_0$

Тогда, преобразование графа к метрической версии его не изменит, следовательно, нам нужно найти минимальный остов на клике из терминальных вершин, где каждое ребро веса 2. Это будет $2\,(|V| - 2)$. С другой стороны, нетрудно понять, что оптимальное поддерево тут будет содержать единственную нетерминальную вершину и все рёбра из неё в терминальные. В таком случае суммарный вес будет $|V| - 1$. Получаем:

$$\frac{2\,(|V| - 2)}{|V| - 1} \to 2,\ |V| \to \infty$$

3.3 Реализация алгоритма

Для проверки алгоритма 2-аппроксимации автор реализовал алгоритм Dreyfus-Wagner, строящий оптимальное поддерево за $O\,(4^t \cdot |V|^2)$ (где t — количество терминальных вершин), а также сформулировал оптимизационную задачу с помощью которой написал ещё один точный алгоритм построения дерева Штейнера на библиотеке ‘pyscipopt‘.

Для построенных алгоритмов написаны тесты, которые проверяют как общие, так и краевые случаи поставленной задачи, а также границы коэффициента аппроксимации.

Посчитано, какую точность даёт построенный приближённый алгоритм, замерено время выполнения всех алгоритмов кроме оптимизационного (он, конечно, даёт всегда оптимальный ответ, но работает несопоставимо долго уже при $|V| \geqslant 20$). Построены соответствующие графики.

Код на C++ получился достаточно большой (около 500 строк), поэтому приведёна только главная часть:

```
int main() {
    ...

    // reading input data
    Graph graph;
    graph.ReadFromList(std::cin);

    // converting into metric task
    MetricGraph metric_graph(graph);
    metric_graph.Convert();

    // finding minimum spanning tree
    MST mst(metric_graph);
    std::vector<Graph::Edge> metric_edges = mst.Find(terminals);

    // expanding our solution from metric back to initial problem
    std::vector<Graph::Edge> edges = ExpandEdgesFromMetricToInitial(
        metric_graph, metric_edges
    );

    // printing the result edges in a subtree
    for (const Graph::Edge& edge : edges) {
        std::cout << edge.from + 1 << ' ' << edge.to + 1 << '\n';
    }

    ...
}
ссылка на алгоритм [3]
```

3.4 Анализ работы

Для сравнения алгоритмов автор решил взять графы $G(n, p)$, так как варьируя параметры n и p по сетке можно получить все возможные графы. Будем обозначать отношение количества терминальных вершин к размеру всего графа за $terminal\_p = \frac{|V_0|}{|V|}$.

Оценка сверху на коэффициент приближения не нарушается. Более того, для сгенерированных входных данных приближённый алгоритм даёт приближение не хуже $\frac{4}{3}$. Также можно заметить, что при такой хорошей точности приближённого алгоритма он работает куда быстрее Dreyfus-Wagner. Это можно наблюдать уже при $n = 30$, $terminal\_p = 0.3$.

Max Approximate Coefficient

p\n	5	10	15	20	30
0.1	1.0	1.0	1.0	1.0	1.219
0.3	1.0	1.0	1.0	1.0	1.250
0.7	1.0	1.0	1.0	1.0	1.333
0.9	1.0	1.0	1.0	1.0	1.250

$terminal\_p = 0.1$

Max Approximate Coefficient

p\n	5	10	15	20	30
0.1	1.0	1.000	1.200	1.235	1.229
0.3	1.0	1.231	1.231	1.267	1.179
0.7	1.0	1.250	1.250	1.222	1.273
0.9	1.0	1.167	1.286	1.200	1.250

$terminal\_p = 0.3$

Average Approximate Coefficient

p\n	5	10	15	20	30
0.1	1.0	1.0	1.0	1.0	1.034
0.3	1.0	1.0	1.0	1.0	1.042
0.7	1.0	1.0	1.0	1.0	1.042
0.9	1.0	1.0	1.0	1.0	1.051

$terminal\_p = 0.1$

Average Approximate Coefficient

p\n	5	10	15	20	30
0.1	1.0	1.000	1.045	1.053	1.050
0.3	1.0	1.015	1.040	1.047	1.055
0.7	1.0	1.027	1.038	1.043	1.058
0.9	1.0	1.017	1.050	1.042	1.056

$terminal\_p = 0.3$

Average Time Complexity (ms)
Dreyfus Wagner

p\n	5	10	15	20	30
0.1	0.01	0.05	0.01	0.05	0.86
0.3	0.02	0.00	0.01	0.17	0.55
0.7	0.01	0.01	0.01	0.17	0.62
0.9	0.02	0.06	0.02	0.14	0.55

$terminal\_p = 0.1$

Average Time Complexity (ms)
Dreyfus Wagner

p\n	5	10	15	20	30
0.1	0.02	0.00	0.48	6.14	245.80
0.3	0.01	0.06	0.93	7.28	252.00
0.7	0.00	0.07	0.88	7.36	252.07
0.9	0.02	0.04	0.89	7.37	249.73

$terminal\_p = 0.3$

Average Time Complexity (ms)
2-Approximation

p\n	5	10	15	20	30
0.1	0.01	0.03	0.02	0.02	0.07
0.3	0.03	0.03	0.01	0.02	0.04
0.7	0.04	0.00	0.01	0.11	0.03
0.9	0.00	0.03	0.01	0.05	0.04

$terminal\_p = 0.1$

Average Time Complexity (ms)
2-Approximation

p\n	5	10	15	20	30
0.1	0.00	0.00	0.00	0.05	0.07
0.3	0.02	0.01	0.01	0.06	0.11
0.7	0.01	0.01	0.06	0.06	0.05
0.9	0.02	0.05	0.02	0.05	0.06

$terminal\_p = 0.3$

4 Вывод

В исследовании была проанализирована одна из ключевых задач в области комбинаторной оптимизации – построение оптимального дерева Штейнера. Мы доказали сложность этой задачи, рассмотрели приближённый метод её решения, и даже получили верхнюю оценку его точности. Также мы провели сравнительный анализ времени работы различных алгоритмов на наборе случайных тестовых задач.

5 Ссылки

1. Karp, *Reducibility Among Combinatorial problems*, 1972. Ссылка
2. Algorithms Lab, *Approximations algorithms for the Steiner Tree Problem and the Traveling Salesperson Problem (TSP)*, 2023. Ссылка
3. Solomon Andryushenko, *Steiner tree 2-approximation polynomial algorithm implementation with tests*, 2024. Ссылка