Остовное дерево минимальной степени

Андрей Рыжов
МФТИ, 2024

30 ноября 2024 г.

Аннотация

В данном отчете рассмотрена задача поиска остовного дерева минимальной степени, то есть остовного дерева, максимальная степень вершины в котором минимальна; доказана **NP**-полнота упрощенной задачи, а также рассмотрены некоторые приближенные алгоритмы решения, один из которых реализован и проанализирован.

1 Теория.

1.1 Постановка задачи.

Дан ненаправленный связный граф $G = (V, E)$, требуется найти его остовное дерево, т. е. подграф, который является деревом и в котором участвуют все вершины, такое что максимальная степень вершины в этом дереве минимально возможна.

1.2 NP-полнота задачи.

Для начала немного упростим задачу и переформулируем её в терминах задач распознавания.

Рассмотрим язык $MDST_k = \{G = (V, E) | G$ – неориентированный граф, в котором существует остовное дерево, степени всех вершин которого не превышают $k\}$, где $k \geqslant 2$.

Теорема 1.1. *Язык $MDST_k$, где $k \geqslant 2$ – **NP**-полный.*

Доказательство. Зафиксируем произвольное $k \geqslant 2$.

Для начала покажем, что язык лежит в **NP**. Построим машину Тьюринга $V(G, T)$, такую что она выполняет следующие действия:

1. Находит размеры графа G (количество вершин) – без ограничения общности будем считать, что граф задаётся как два числа (кол-во вершин n и ребёр m) и матрица смежности.
2. Проверяет, что для каждой вершины из G существует вершина с таким номером в T.
3. Проверяет, что для каждой вершины из T существует вершина с таким номером в G.
4. Проверяет, что граф T – связен.
5. Проверяет, что в графе T ровно $n - 1$ ребро.
6. Проверяет, что ребра, которые есть в графе T, есть и в графе G.
7. Проверяет, что степень каждой вершины в T не превышает k.

Если хотя бы одна проверка провалена, то алгоритм возвращает отрицательный результат, иначе – положительный.

Очевидно, что все эти проверки – полиномиальные от $|T| + |G|$ (реализуются за $O(|G|^2)$ как стандартные алгоритмические задачи, а значит с некоторым полиномиальным замедлением реализуются и на машине Тьюринга), то есть занимают $poly(|G|)$ времени (поскольку $|G| \geqslant |T|$, иначе работу можно сразу завершить, отвергнув такую пару).

Понятно, что если граф T – остовное дерево для графа G, то пара (G, T) пройдёт все проверки, поскольку каждая из них является необходимым условием. Обратно, если T удовлетворяет условиям, то:

1. T – дерево, поскольку в нём $n - 1$ ребро и n вершин, а сам граф – связный.
2. T – остовное дерево, поскольку множество его вершин совпадает с множеством вершин G, а рёбра являются подмножеством рёбер графа G.
3. Наконец, максимальная степень вершины в графе T не превышает k (проверяется непосредственно).

Значит, $\exists T : V(G,T) = 1 \Leftrightarrow G \in MDST_k$, откуда $MDST_k \in \mathbf{NP}$.

Теперь покажем **NP**-трудность, что в совокупности с предыдущим пунктом даст **NP**-полноту.

Сведём известную **NP**-полную задачу к нашей, а именно покажем, что язык UHAMPATH (состоящий из неориентированных графов, в которых существует гамильтонов путь) сводится к нашему.

Пусть нам дан неориентированный граф G. Построим по нему полиномиальным преобразованием новый граф G', а именно: для каждой вершины заведём $k-2$ новые вершины и соединим их со старой. Понятно, что такое преобразование вычисляется за полиномиальное время от размера G (k – константа в рамках задачи).

Покажем, что $G \in \mathsf{UHAMPATH} \Leftrightarrow G' \in MDST_k$.

Пусть $G \in \mathsf{UHAMPATH}$.

Заметим, что в гамильтоновом пути степень каждой вершины не превышает 2 (войти в вершину и выйти из неё мы можем лишь однажды). Тогда, добавляя к гамильтонову пути по $k-2$ новых вершины для каждой старой (а также ребра, связывающие их с "оригиналом"), получаем остовное дерево для G', причём степень каждой вершины в нём не превышает $2 + (k-2) = k$, то есть $G' \in MDST_k$.

Обратно, пусть $G' \in MDST_k$.

Пусть P – остовное дерево степени не выше k в G', а S – его подграф, индуцируемый вершинами из G. Заметим, что:

1. Степень каждой вершины в S не превышает 2 (т. к. в P она не превышала k и для каждой вершины удалили её $k-2$ клонов-соседей).
2. В S находятся все вершины из G и только они (по построению).
3. В S нет циклов (поскольку они отсутствуют в P).
4. Наконец, граф S – связный: пусть не так, тогда воспользуемся связностью P и заметим, что из нашего предположения вытекает, что существуют две вершины в S, которые были соединены путём, проходящим через одну из новых (фиктивных) вершин. Но так быть не могло: каждая фиктивная вершина связана только с её родителем, к которому она была присоединена, и ни с кем больше. Значит, предположение неверно и S – связный граф.

Отсюда следует, что S – остовное дерево в G; из оценки на степень вершин заключаем, что S – гамильтонов путь (остовное дерево, степень каждой вершины которого не больше 2), то есть $G \in \mathsf{UHAMPATH}$, что и требовалось.

□

1.3 Приближенные полиномиальные алгоритмы.

Поскольку даже упрощенная задача **NP**-полна, на данный момент неизвестен алгоритм, решающий её за полиномиальное время. Тем не менее, известны некоторые полиномиальные алгоритмы, дающие приближенное решение (здесь T – получаемое решение, $\Delta(T)$ – максимальная степень вершины в нём, $\Delta^*(G)$ – настоящий ответ для графа G):

1. Алгоритм, использующий линейное программирование, рассмотренный в [1], позволяет решить эту (и даже более общую задачу – когда у рёбер есть веса) задачу с точностью $\Delta(T) \leqslant \Delta^*(G) + k$ за полиномиальное время, где k – различное число стоимостей рёбер в остовном дереве для G.
2. Алгоритм, рассмотренный в [2], работающий за $O(n^2)$ и приближающий ответ с точностью $\Delta(T) \leqslant 2\Delta^*(G) + \log_2(n)$.
3. Алгоритм, также рассмотренный в [2], работающий за $O(mn\log(n)\alpha(n))$ и приближающий ответ с точностью $\Delta(T) \leqslant \Delta^*(G) + 1$.

И другие. В данной работе сосредоточимся на последнем алгоритме.

2 Полиномиальный алгоритм с точностью $\Delta(T) \leqslant \Delta^*(G) + 1$.

2.1 Описание работы.

Алгоритм получает на вход граф $G = (V, E)$ в виде списка вершин и списка рёбер (граф неориентированный). Затем:

1. Алгоритм проверяет, что граф связен, и в противном случае сразу завершает работу, поскольку в этом случае остовное дерево не существует.
2. Алгоритм находит произвольное остовное дерево (например, с помощью алгоритма **BFS** за линейное от размера графа время).

3. Алгоритм выполняет итеративный процесс, который на каждой итерации либо уменьшает число вершин в T, имеющих максимальную степень, либо завершает работу.

Итеративный процесс состоит из следующих действий на каждой итерации:

1. Из остовного дерева T удаляются все вершины с текущей максимальной степенью k, полученный граф T' конденсируется по компонентам связности.
2. Алгоритм перебирает все рёбра из графа G и пробует улучшить остовное дерево, а именно ищет такие ребра между двумя компонентами связности, что для каждой из двух его вершин u_1, u_2 выполнено одно из двух условий: либо степень вершины меньше $k-1$, либо степень вершины равна $k-1$ и можно её понизить, то есть существует некоторое ребро (v_1, v_2) в компоненте связности вершины u_i (ребро из G), такое что путь от v_1 до v_2 в T проходит через u_i и при этом степени вершин v_1 и v_2 строго меньше $k-1$.
3. Если такое ребро не найдено, то работа завершена.
4. Если такое ребро найдено, то мы добавляем его в T и удаляем некоторое ребро из T как бы взамен, а именно удаляем ребро, которое ранее соединяло компоненты связности u_1 и u_2.

Покажем теперь, что данный алгоритм корректен. Воспользуемся в том числе материалами статьи [2].

2.2 Доказательство корректности.

В основе доказательства корректности работы алгоритма лежит следующая теорема.

Теорема 2.1. *Пусть T – произвольное остовное дерево в графе $G = (V, E)$ и k – максимальная степень вершины в T. Положим $S = \{v \in V | deg_T(v) = k\}$, $B \subseteq \{v \in V | deg_T(v) = k-1\}$. Рассмотрим лес F из деревьев на вершинах $V \setminus (S \cup B)$. Тогда если G удовлетворяет условию, что между деревьями F нет рёбер, то $k \leqslant \Delta^*(G) + 1$.*

Доказательство. Поскольку между деревьями из леса F нет рёбер из G, то единственный способ, которым они могут быть соединены для получения остовного дерева – через вершины из $S \cup B$. Сконденсируем лес F в новое множество вершин K.

Рассмотрим произвольное остовное дерево в графе G и заметим, что оно является связным графом на $K \cup S \cup B$, возможно, с петлями. Тогда в нём хотя бы $|K| + |S| + |B| - 1$ ребро (все они так или иначе соединены с вершинами из $S \cup B$). В то же время, если вершина имела степень x в графе, то она соединяла x компонент связности, то есть в F по крайней мере $k|S| + (k-1)|B| - 2(|S| + |B| - 1)$ компонент связности, поскольку мы могли посчитать лишние компоненты на вершинах из $S \cup B$, однако ошибка не превышает суммарную степень всех вершин в подграфе T на $S \cup B$, то есть не более $2(|S| + |B| - 1)$ (так как они образуют подмножество дерева), из чего и следует требуемое.

Тогда имеем, что средняя степень вершины в $S \cup B$ в произвольном остовном дереве по крайней мере $\frac{|S|+|B|+|K|-1}{|S|+|B|} \geqslant \frac{k|S|+(k-1)|B|-2(|S|+|B|-1)+|S|+|B|-1}{|S|+|B|} = k - 1 - \frac{|B|-1}{|S|+|B|}$. Отсюда, так как $\frac{|B|-1}{|S|+|B|} < 1$, получаем, что максимальная степень вершины в $S \cup B$ хотя бы $k-1$, а значит и $\Delta^*(G) \geqslant k-1$, или же $k \leqslant \Delta^*(G) + 1$, что и требовалось.

□

Теперь понятна корректность алгоритма. Пусть мы не смогли улучшить ответ на очередной итерации и завершили работу. Обозначим множество вершин степени k за S_k.

Покажем, что тогда граф подходит под условия теоремы. Поскольку алгоритм остановился, то нет рёбер между компонентами связности в $V \setminus S_k$, таких что их добавление может улучшить ситуацию (уменьшить число вершин степени k), то есть нет рёбер между такими парами вершин из разных компонент, что каждая из них либо имеет степень меньше $k-1$, либо имеет степень $k-1$ и степень может быть локально понижена (то есть с использованием ребра из той же компоненты связности).

Рассмотрим произвольное такое ребро (u_1, u_2). Тогда без ограничения общности хотя бы одна вершина (пусть это u_1) имеет степень $k-1$, и в её компоненте связности нет ребра (v_1, v_2) (в G), такого что путь между v_1 и v_2 проходит через u_1 и степени вершин v_1 и v_2 меньше $k-1$, или, иначе говоря, добавление любого ребра в компоненте связности вершины u_1 образует цикл, в котором есть вершина степени k. Обозначим множество всех таких вершин за S_{k-1}. Тогда очевидно, что такая ситуация подходит под условие теоремы: если между компонентами в $V \setminus (S_k \cup S_{k-1})$ было бы ребро, то оно бы либо улучшало ситуацию во всём графе напрямую (уменьшая число вершин степени k), либо позволяло бы понизить степень одной из вершин степени $k-1$ в пределах одной компоненты (поскольку те из них, что делать этого не позволяют, уже отсечены, так как

хотя бы один из их концов лежит в S_{k-1}). Значит, полученное остовное дерево имеет степень, отличающуюся от оптимальной не более чем на 1, что и требовалось.

2.3 Время работы алгоритма.

Нетрудно заметить, что алгоритм работает за полиномиальное время: поиск произвольного остовного дерева может быть выполнен за линию каким-либо обходом; на каждой итерации мы удаляем какие-то вершины, конденсируем граф и перебираем все рёбра, проверяя некоторое условие за $O(1)$, то есть каждая итерация также линейна. Наконец, понятно, что так как каждый раз мы уменьшаем число вершин максимальной степени в T на 1, то всего может быть выполнено не более n^2 итераций. Таким образом, алгоритм полиномиален. С использованием различных оптимизаций (в первую очередь при конденсации графа), можно также реализовать алгоритм, время работы которого оценивается как $O(mn\log(n)\alpha(n))$.

3 Практика.

3.1 Реализация алгоритма.

Поскольку не требуется реализовать оптимальный алгоритм, в рамках данной работы реализован вариант, работающий за полиномиальное, но чуть худшее время, чем минимально возможное, отмеченное выше. Код реализации может быть найден в репозитории: MDST-solver.

3.2 Анализ работы алгоритма на различных входных данных.

To be continued...

Список литературы

[1] R. Ravi, Mohit Singh; Delegate and Conquer: An LP-Based Approximation Algorithm for Minimum Degree MSTs. — 2006. — URL: `https://link.springer.com/chapter/10.1007/11786986_16`.

[2] Martin Fürer, Balaji Raghavachari; Approximating the Minimum Degree Spanning Tree to within One from the Optimal Degree. — 1992. — URL: `https://www.researchgate.net/publication/220779201_Approximating_the_Minimum_Degree_Spanning_Tree_to_Within_One_from_the_Optimal_Degree`.