Поиск максимальной клики в графе

Морозова Анастасия Валентиновна Б05-223

21 мая 2024 г.

1 Описание задачи

Задача о клике принадлежит классу NP-полных задач в области теории графов. Кликой в графе (неориентированном) называется подмножество вершин, каждые две из которых соединены ребром графа. Соответственно задача состоит в том, чтобы найти максимальную клику в заданном неориентированном графе.

2 NP-полнотая задачи

Мы знаем, что задача о независимом множестве вершин NP-полная. Чтобы существовала клика размера k, нужно чтобы существовало независимое множество размера >= k в графе, являющимся дополнением к данному. Следовательно, из NP-полноты задачи о независимом множестве следует NP-полнота данной.

3 Решение задачи

3.1 Алгоритмы

Существует несколько алгоритмов решения этой задачи. Полный перебор всех возможных подграфов размера k с проверкой того, является ли хотя бы один из них полным, не эффективный.

Другой алгоритм работает так, что две клики размера n и m собираются в одну клику размера n+m, реализовать такое можно с помощью динамического программирования. Алгоритм завершается, как только ни одного слияния больше произвести нельзя. Однако алгоритм является эвристическим, мы не можем гарантировать что ответ будет правильным. При этом в хорошем случае можем ожидать линейное время работы. Существует алгоритм Брона-Кербоша, метод ветвей и границ для поиска всех клик, который я и хочу реализовала. Этот алгоритм ищет все возможные клики в неориентированном графе. Он был разработан математиками Броном и Кербошем и до сих пор является одним из самых эффективных алгоритмов поиска клик.

3.2 Описание алгоритма

Алгоритм строит клики (полные подграфы) в графе, полагаясь на тот факт, что каждая клика уже является максимальной по включению. Он начинает с одиночной вершины (представляющей собой полный подграф) и на каждом шаге пытается расширить текущий полный подграф, выбирая вершины из списка кандидатов. Для повышения эффективности алгоритм использует дополнительный список, в который помещает использованные вершины, чтобы исключить неверные варианты, которые не приведут к созданию клики. Алгоритм использует три набора вершин для поиска клик (полных подграфов) в графе:

- множество, содержащее полный подграф на каждом шаге поиска
- множество вершин, которые могут быть добавлены в первое множетсво для расширения подграфа
- множество вершин, которые уже были использованы для расширения первого множества на предыдущих шагах поиска

Используя эти наборы, алгоритм ищет клики, начиная с одиночной вершины и постепенно добавляя вершины из кандидатов, которые соответствуют критериям формирования клики. Последнее множество помогает избежать повторного использования вершин.

3.3 Доказательство ассимптотики

Чтобы оценить, как в худшем случае работает алгоритм, докажем утверждение:

<u>Утв.</u> Обозначим через $f(n)$ max количество max клик

$$n \geq 2 \Rightarrow f(n) = \begin{cases} 3^{n/3}, & n \equiv_3 0 \\ 4 \cdot 3^{[n/3]-1}, & n \equiv_3 1 \\ 2 \cdot 3^{[n/3]}, & n \equiv_3 2 \end{cases}$$

Δ Нетрудно проверить при $n=2$ (клика разм. 2), 3, 4 анал-но.

Тогда стоит рассмотреть графы при $n \geq 5$ (пусть связные) и обозначим кол-во клик через $c(G)$. Смежные вершины вершины x обозначим $\Gamma(x)$.

$\exists$ $\alpha(x)$ графов сод. в $\Gamma(x)$, явл-ся max по отн. к $G/\{x\}$.

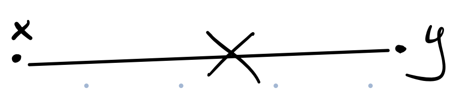
$\exists$ $\beta(x)$ графов сод. в $\Gamma(x)$, явл-ся max по отн. к $\Gamma(x)$ но не $G/\{x\}$

$$\Rightarrow C(G/\{x\}) = C(G) - \beta(x) \qquad \beta(x,y) = \beta(y,x)$$

Тогда обозначим за $\chi(x)$ число клик в G, содержащих x.

Т.к. $\alpha(x)$ и $\beta(x)$ доп. друг друга и не пересекаются то очевидно, что $\chi(x) = \alpha(x) + \beta(x)$.

Рассмотрим 2 несвязанные вершины в графе G: x и y.



$\exists$ $G(x,y)$ обозначает такой граф, что связанные с x ребра удаляются и заменяются ребрами, соед. x с каждой вершиной $\Gamma(y)$

$$c(G(x,y)) = c(G) + \chi(y) - \chi(x) + \alpha(x)$$

Пусть G - любой граф, у к-го $n \geq 5$ вершин, а также max кол-во клик. G связен и ни одна вершина не связана с каждой оставшейся вершиной.

$\chi(y) > \chi(x) \Rightarrow G(x;y)$ был бы более выгодным по кол-ву клик, а потому в нашем графе G $\chi(y) = \chi(y)$ $\forall x, y : \nexists e(x,y)$ - ребро, соед. x и y.

Поэтому $\alpha(x) = 0$ $\forall x \in G$. $G^{(1)} = G$.

Возьмем произвольную вершину $x \in G$ и обозначим за a, b, c, d, e, f не связанные с ней. $G^{(2)} = G(a,x)$. Заменим $G^{(1)}$ на $G^{(2)}$ потому что это не влияет на кол-во и размер max клики. Заменим далее $G^{(2)}$ на $G^{(3)}$ и т.д. Теперь получим граф с такими же св-вами, но $x, a, \dots f$ в нем не связаны между собой, при этом связаны с остальными. Теперь применяем ту же процедуру к вершине y в $\Gamma(x)$. Получим итоговый граф $\hat{G}$, т.ч. мы можем разделить вершины на непересекающиеся мн-ва по правилу: $\exists e(x,y) \Leftrightarrow x, y$ не лежат в 1 множестве

Если эти подмножества содержат $j_1, \dots j_t$, где $j_1 + \dots + j_t = n$, то

$$c(\hat{G}) = j_1 \cdots j_t$$

$c(\hat{G})$ достигает max значения, если max количество непересекающихся мн-в содержат 3 вершины, а оставшиеся могут иметь по 2 если остаток равен 2, а иначе 4 если остаток 1.

3.4 Тестовые запуски

Алгоритм достаточно простой, за клику мы считаем подграф изначального графа, в котором все вершины соединены между собой, но размера >= 3. Поэтому на тестовом запуске с одним ребром или одной вершиной в графе выдает ответ []:

```
if __name__ == '__main__':
    testcase = [('a', 'b'), ('b', 'a')]
    printResult(testcase)
```

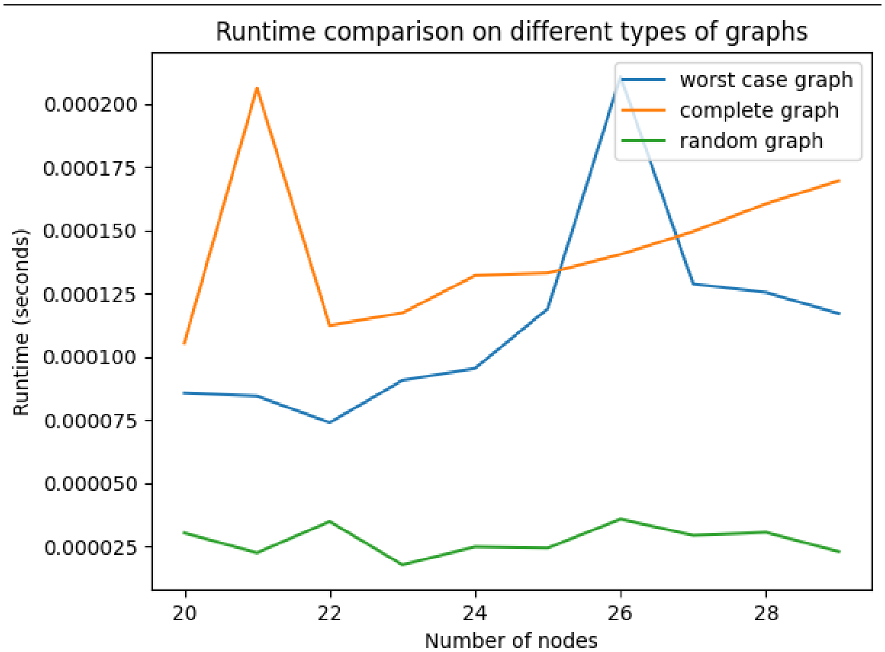
1) Написаны тесты для простых графов

2) Написаны тесты для случайных графов G(n, p) где вершин от 1 до 20 (случайная величина), и вероятность ребра от 0.4 до 0.8 (равномерно распределенная величина)

3) Отдельно также написаны тесты для дистанционного и планарного графа

4) А также замерила время работы алгоритма на случайных графах (где $[0;3^{n/3}]$ максимальных клик) и время работы на полных графах на таком же количестве вершин (где максимальная клика всегда 1 размера n). На выборке из 1000 запусков в среднем полный граф (где всего 1 максимальная клика) обрабатывается алгоритмом быстрее чем случайный граф с числом наибольших клик в $>= 3^{n/7}$ на 0.0015 секунд. Это число практически не меняется в зависимости от запуска. Так как худшее значение асимптотики эвристического алгоритма реализуется при примерно $3^{n/3}$ кликах, то задержка только будет увеличиваться. Среднее время работы на случайных графах: 0.0012011096477508537 секунд. То есть на 'плохих' графах алгоритм работает почти в 2 раза медленнее.

При этом полный граф не то же самое что случайный граф. Полный граф тяжело обрабатывать из-за количества вершин в нем. Для более полной картины посмотрим на сравнение с дейтсвительно случайными графами.



Как видно из результатов, время работы алгоритма Брона-Кербоша на случайных графах значительно меньше $O(3^{n/3})$, а так же лучше чем полных графов или графов с большим количеством наибольших клик. Это связано с тем, что случайные графы обычно имеют гораздо меньше клик, чем граф наихудшего случая или полный.

4 Вывод

Алгоритм Брона-Кербоша — это эффективный алгоритм поиска всех максимальных клик в графе. Он оптимален в том смысле, что находит все максимальные клики и делает это за время $O(3^{n/3})$, где n — количество вершин в графе. Однако на случайных графах алгоритм Брона-Кербоша обычно занимает гораздо меньше времени, чем $O(3^{n/3})$, поскольку случайные графы обычно имеют гораздо меньше максимальных клик, чем граф наихудшего случая.