Поиск максимальной клики в графе. Алгоритм Брона — Кербоша

Быстров Сергей, Б05-220

Содержание

1 Введение

В данном проекте будем рассматривать задачу поиска максимальной клики в графе.

Определение 1 *Клика, которая не содержится в клике большего размера, называется максимальной кликой.*

Задача о поиске максимальных клик принадлежит классу NP-полных задач. В рамках данного проекта рассмотрим алгоритм Брона — Кербоша, имеющий асимптотику

$$\mathcal{O}(\#\text{количество клик максимального размера}) = \mathcal{O}(3^{\frac{n}{3}})$$

где n - количество вершин в графе.

Алгоритм был разработан голландскими математиками Броном и Кербошем в 1973 году, и несмотря на экспоненциальную асимптотику хорошо себя показывает на большинстве графов (особенно его улучшения, которые мы также рассмотрим), поэтому и представляет собой большой интерес.

2 Теоретическая часть

2.1 Базовая версия алгоритма

Алгоритм строит клики в графе итерационно, полагаясь на тот факт, что каждая клика уже является максимальной по включению. Он начинает с одиночной вершины и на каждом шаге пытается расширить текущий полный подграф, выбирая вершины из *списка кандидатов*. Отдельным множеством алгоритм передаются вершины, которые уже точно не приведут к созданию клики. Таким образом в алгоритме задействованы три множества:

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

Таким образом, базовая версия алгоритм выглядит следующим образом:

Algorithm 1 Алгоритм Брона — Кербоша

```
1: procedure BronKerbosch1(R, P, X)
2:     if P и X обе пусты then
3:         return R как максимальную клику
4:     end if
5:     for каждая вершина v в P do
6:         BronKerbosch1(R ∪ {v}, P ∩ N(v), X ∩ N(v))
7:         P := P \ {v}
8:         X := X ∪ {v}
9:     end for
10: end procedure
```

Базовая форма алгоритма, описанная выше, неэффективна в случае графов с множеством немаксимальных клик: он делает рекурсивный вызов для каждой клики, и неважно максимальной или нет. Следующие две вариации этого алгоритма избавляются от лишних рекурсий. Первый выбирает "поворотную вершину"из P, соседи которой не будут проверяться на нахождение в максимальной клики, второй задает специальный порядок на вершинах, обход которого позволяет убрать лишние итерации.

2.2 Первое улучшение: with pivoting

Рассмотрим подробнее первое улучшение: выбор опорной точки. Чтобы сэкономить время и позволить алгоритму быстрее возвращаться из ситуаций, которые не содержат максимальных клик, Брон и Кербош представили [2] вариант алгоритма, включающий "поворотную вершину"u, выбранную из P. Соседи этого опорного элемента не проверяются рекурсивно. Любая максимальная клика, потенциально найденная в тестах соседей u, также будет найдена при тестировании u или одного из его несоседей, поскольку хотя бы один из них всегда будет частью этой максимальной клики. В противном случае только соседи u были бы частью этой максимальной клики, что позволяло бы увеличить ее путем добавления к ней u что противоречит тому, что эта клика является максимальной. Следовательно, только u и ее не-соседей необходимо проверять в качестве выбора вершины v, которая добавляется к R при каждом рекурсивном вызове алгоритма.

Algorithm 2 Алгоритм Брона — Кербоша с опорной вершиной (Bron-Kerbosch2)

```
1: procedure BronKerbosch2(R, P, X)
2:     if P = ∅ and X = ∅ then
3:         report R                                  ▷ Нашли максимальную клику
4:     end if
5:     Выбрать опорную вершину u из P ∪ X
6:     for each v ∈ P \ N(u) do
7:         BronKerbosch2(R ∪ {v}, P ∩ N(v), X ∩ N(v))
8:         P ← P \ {v}
9:         X ← X ∪ {v}
10:    end for
11: end procedure
```

2.3 Второе улучшение: with vertex ordering

Альтернативный метод улучшения базовой версии алгоритма Брона — Кербоша заключается в отказе от выбора опорной вершины на самом внешнем уровне рекурсии. Вместо этого тщательно выбирается порядок рекурсивных вызовов с целью минимизации размеров множеств P кандидатов в вершины на каждом уровне рекурсии.

Определение 2 *Вырожденность графа G определяется как минимальное число d, такое что любой подграф графа G содержит вершину со степенью не больше d.*

Для любого графа можно построить порядок вырожденности — порядок вершин, при котором каждая вершина имеет не более d соседей, идущих позже в этом порядке. Такой порядок можно найти за линейное время, последовательно выбирая вершину с минимальной степенью среди оставшихся вершин.

Если порядок вершин v, по которым выполняется цикл в алгоритме Брона — Кербоша, соответствует порядку вырожденности, то множество P (кандидатов в вершины для текущего рекурсивного вызова) будет гарантированно иметь размер не больше d (соседи v, которые идут позже в порядке). Множество X (исключённых вершин) будет содержать всех соседей v, идущих раньше, и может быть гораздо больше d. На уровнях рекурсии ниже верхнего уровня можно по-прежнему использовать версию алгоритма с опорной вершиной.

Algorithm 3 Алгоритм Брона — Кербоша с порядком вырожденности (Bron-Kerbosch3)

```
1: procedure BronKerbosch3(G)
2:     P ← V(G)
3:     R ← ∅, X ← ∅
4:     for each вершина v в порядке вырожденности графа G do
5:         BronKerbosch2({v}, P ∩ N(v), X ∩ N(v))
6:         P ← P \ {v}
7:         X ← X ∪ {v}
8:     end for
9: end procedure
```

2.4 Корректность

1. Алгоритм распознает только максимальные клики. Действительно, мы начинаем в вершине - клики. На каждом из следующих шагов мы получаем только клики, так как во множестве P остаются только вершины, имеющие ребра со всеми вершинами из множества X

2. Мы находим все максимальные клики. Действительно, так как мы начинаем во всех вершинах, то в том числе и в вершине, находящейся в рассматриваемой клики. На каждом этапе, мы получаем клику, завершаемся когда не остается вершин, соединенных со всеми вершинами из нашего множества X, то есть когда клику нельзя расширить, то есть когда достигли максимальную клику.

2.5 Асимптотика

В случае удачно выбранных pivot-вершин для второго варианта алгоритма количество итераций ограничено количеством максимальных клик. Аналогична ситуация и для третьего варианта. Эти нетривиальные факты приведены и доказываются в [2] и [1] соответственно. Мы же покажем, что число максимальных клик есть $\mathcal{O}(3^{\frac{n}{3}})$ (теорема Moon и Moser (1965) [3]), тем самым покажем, что асимптотика улучшенных версий алгоритм $\mathcal{O}(3^{\frac{n}{3}})$

Jбозначим через $f(n)$ количество максимальных клик $n \geq 2$, тогда

$$f(n) = \begin{cases} 3^{\frac{n}{3}} & n \equiv 0 \mod 3 \\ 4.3^{\frac{n}{3}-1} & n \equiv 1 \mod 3 \\ 2.3^{\frac{n}{3}} & n \equiv 2 \mod 3 \end{cases}$$

Нетрудно проверить это при $n = 2$ (клика размера 2), для 3, 4 аналогично. Тогда стоит рассмотреть графы при $n >= 5$ (допустим, граф связен) и обозначим количество клик в графе черер $c(G)$. Смежные вершины обозначим через $\Gamma(x)$.

Через $\alpha(x)$ обозначим количество графов, содержащихся в $\Gamma(x)$, являющихся максимальными по отношению к $G/$x.

Через $\beta(x)$ обозначим количество графов, содержащихся в $\Gamma(x)$, являющихся максимальными по отношению к $\Gamma(x)$, но не $G/$x. Заметим, что $\beta(x,y) = \beta(y,x)$

Тогда также очевидно, что $c(G/x) = c(G) - \beta(x)$. Обозначим за $\chi(x)$ число клик в G, содержащих x.

Так как $\alpha(x)$ и $\beta(x)$ дополняют друг друга и не пересекаются, то $\chi(x) = \alpha(x) + \beta(x)$.

Рассмотрим две несвязанные вершины в графе G: x и y. Обозначим через G(x,y) такой граф, что связанные с x ребра удаляются и заменяются ребрами, соединяющие x с каждой вершиной $\Gamma(y)$.

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

Пусть G - любой граф, у которого $n >= 5$ вершин, а также максимальное количество клик. G связан, ни одна вершина не связана с каждой оставшейся.

$\chi(y) > \chi(x)$ => G(x,y) был бы более выгодным по количеству клик, а потому в нашем графе G: $\chi(y) = \chi(x)$. $\forall x \in G \alpha(x) = 0$. $G^1 = G$ (это обозначение, позже мы используем G с индексами).

Возьмем произвольную вершину x $\in Ga, b, c, d, e, f$.G$^2 = G(a, x)$. Заменим теперь G^1 на G^2, потому что это не влияет на количество и размер максимальной клики. Заменяем далее G^2 на G^3 и так далее. После всех замен получим граф с теми же свойствами, но x, a, ... f не будут связаны между собой, при этой связаны с остальными. Теперь применим ту же процедуру к вершине y в $\Gamma(x)$. Получили итоговый граф G, такой что мы можем разделить вершины на непересекающиеся множества по правилу: $\exists e(x,y) <=> x, y.\text{j}_1$, ... j_t, где $j_1 + j_2 + ... + j_t = n$, то $c(G) = j_1 * j_2 * ... * j_t$.

Далее применяя метод Лагранжиана находим, что максимум достигается в случае, когда $j_1 = j_2 = \cdots = j_t$. Тогда рассматривая n, по модулю 3, получаем оценку $\mathcal{O}(3^{\frac{n}{3}})$.

Оценка очевидно достигается для графа Турана $T(n, n/3)$.

Определение 3 ***Граф Турана*** *$T(n,r)$ — это граф, образованный разложением n вершин на r подмножеств, с как можно близким размером, и вершины в этом графе соединены ребром, если они принадлежат разным подмножествам*

Мы из каждого множества можем выбрать по одной вершине, поэтому все максимальные клики имеют размер 3, а всего их $3^{\frac{n}{3}}$.

3 Практическая часть

3.1 Описание тест-кейсов

Мы сравним два вида алгоритма: оригинальный и with vertex ordering, причем во втором случае для рекурсивных алгоритмов будем запускать алгоритм with pivoting. Сравним эффективность этих алгоритмов на примере случайных графов, на полных графах, на графах Турана. Такое выбор сделан по следующим причинам: случайные графы позволяют эмулировать реальные графы, которые можно встретить в жизни. На полных графах мы хотим увидеть неэффективность исходного алгоритма (а именно, то как он обходит все клики в графе, что в нашем случае есть все подмножества) и эффективность алгоритма with vertex ordering, который оценивается сверху количеством максимальных клик (в нашем случае единицей). Графы Турана нам интересны по той причине, что они являются примером достижения наихудшей оценки.

3.2 Случайные графы

Рассмотрим случайные графы с количеством вершин от 50 до 300 и вероятностями возникновения ребра 0.2, 0.3, 0.4 (мы не хотим рассматривать слишком большие значения p, так как а) будет долго работать, б) рассмотрим полные графы в отдельном тесте. Далее представлены графики зависимости времени от количества вершин 1, 2, 3:

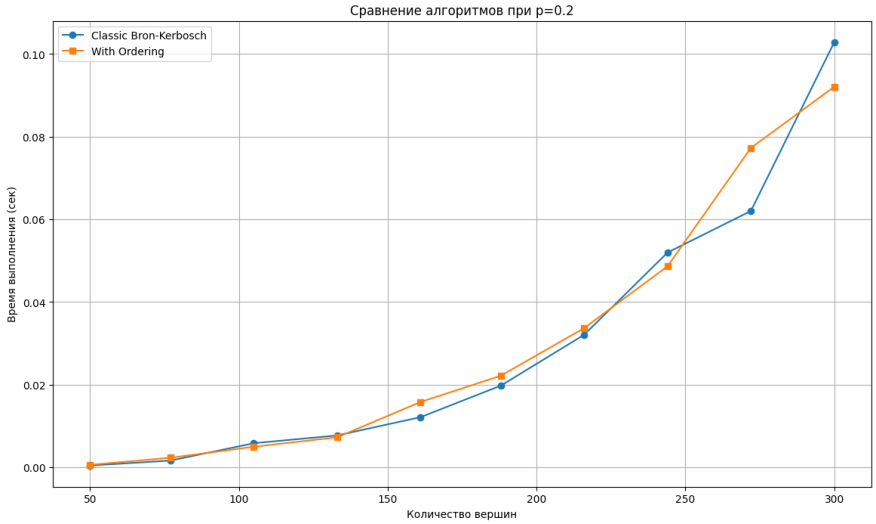

Рис. 1: p=0.2

Видим, что алгоритмы показывают себя практически одинаково. Связано это с тем, что вариант with ordering лучше всего показывает себя в случае большого числа немаксимальных клик, ожидать на случайных графах этого не стоит.

3.3 Полные графы

Рассмотрим полные графы с небольшим числом вершин (так как оригинальный алгоритм плохо справляется с такими графами) 4

Здесь мы видим главный недостаток исходного алгоритм. В то время как улучшенная версия отлично справляется (в полных графах только одна максимальная клика), оригинальный алгоритм перебирает все подмножества, из-за чего страдает.

Видим, что время, затраченное улучшенной версией 5, ничтожно.

3.4 Графы Турана

Рассматриваем графы Турана $T(n, \frac{n}{3})$ 6.

Видим, что в среднем улучшенная версия показывает себя лучше, но разрыв не настолько большой. Связано это с тем, что такие графы Турана - пример, когда наихудшая оценка на асимптотику достигается.

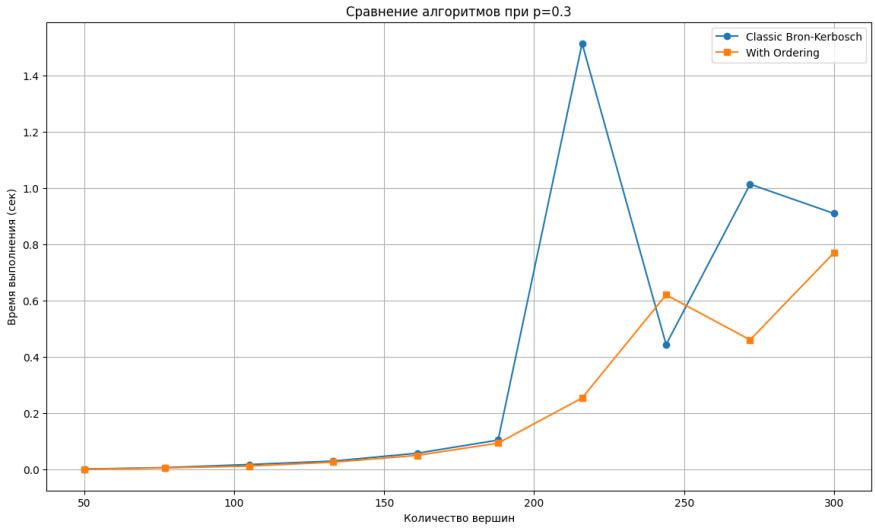

Рис. 2: p=0.3

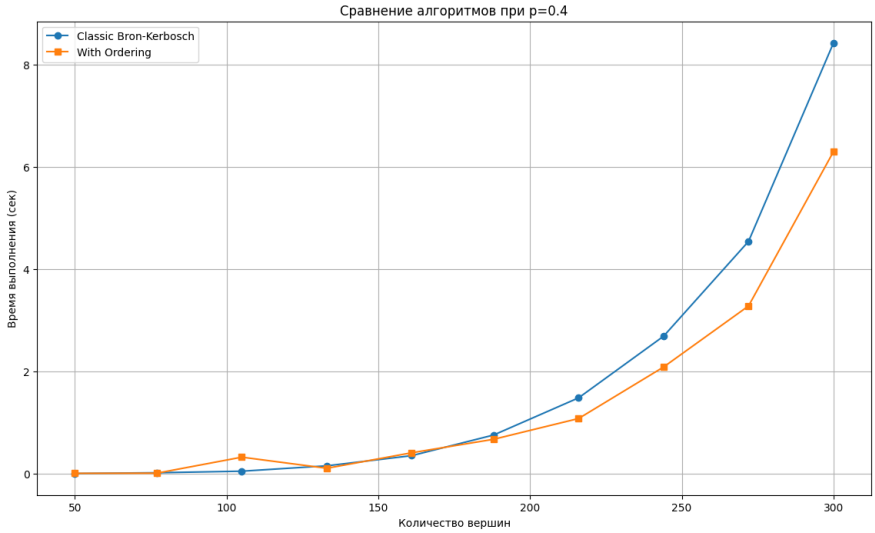

Рис. 3: p=0.4

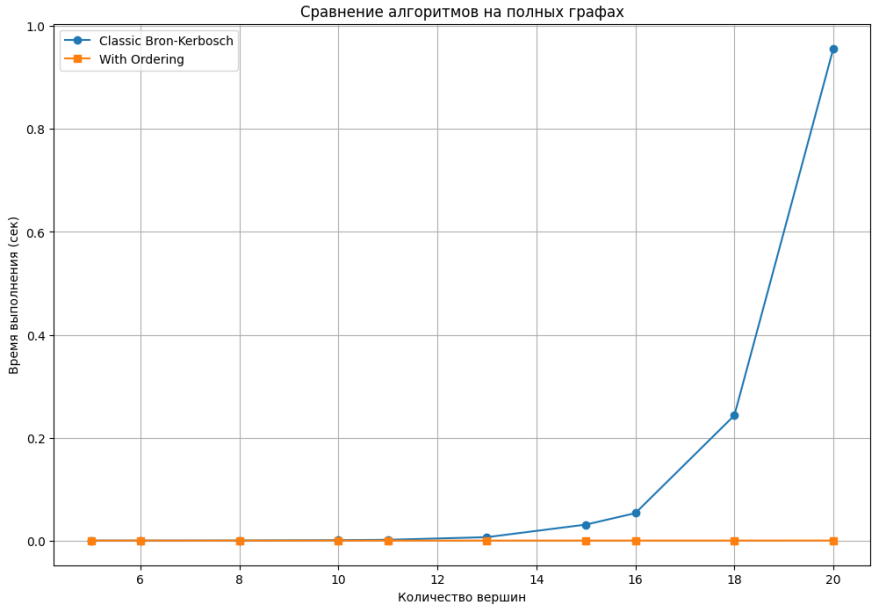

Рис. 4: Сравнение алгоритмов на полных графах

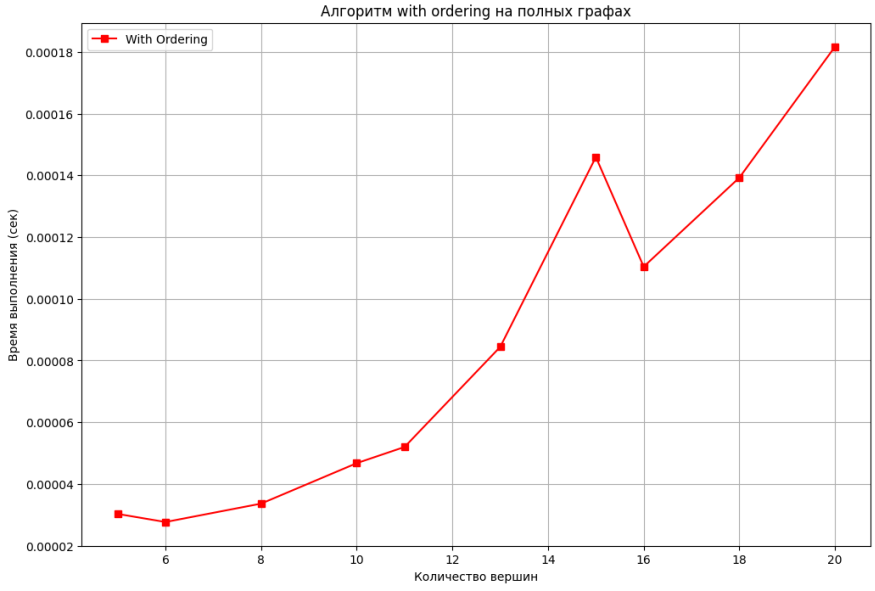

Рис. 5: Улучшенная версия алгоритма на полных графах

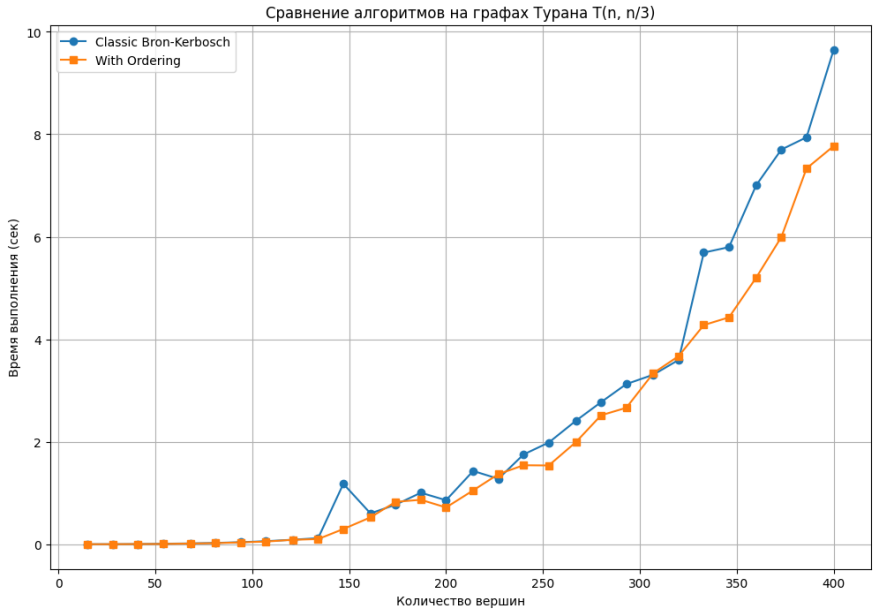

Рис. 6: Сравнение алгоритмов на графах Турана

4 Вывод

В работе мы рассмотрели задачу поиска максимальных клик в графе, а конкретно алгоритм Брона-Кербоша. Базовая версия может быть неэффективной на больших графах, поэтому расмотрели несколько стандартных его улучшений.

Первое улучшение — использование "поворотной вершины которое позволяет избежать лишних проверок и ускорить поиск. Второе улучшение — это упорядочивание вершин, что позволяет минимизировать размер множества кандидатов на каждом шаге и таким образом уменьшить количество итераций.

На практике мы сравнили 2 версии алгоритма на различных типах графов: случайных графах, полных графах и графах Турана. Эксперименты показали, что улучшенная версия, комбинирующая два улучшений работает в среднем быстрее, особенно быстрее на полных графах, где оригинальный алгоритм испытывает большие проблемы из-за перебора всех подмножеств.

Список литературы

[1] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. *The worst-case time complexity for generating all maximal cliques and computational experiments.* The University of Electro-Communications, Department of Information and Communication Engineering, Chofu, Tokyo, Japan, 2006.

[2] Coen Bron and Joep Kerbosch. *Algorithm 457: finding all cliques of an undirected graph.* Communications of the ACM, 1973.

[3] J. W. Moon and L. Moser. *On cliques in graphs.* Israel Journal of Mathematics, 1965.