Поиск максимальной клики в графе

Морозова Анастасия Валентиновна Б05-223

Содержание

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.

Определение 1 *Будем рассматривать простые (без петель и кратных ребер) неориентированные графы G = (V, E). Кликой Q (или полным подграфом) графа G называется такое подмножество его вершин, в котором любые две вершины соединены ребром.*

Определение 2 *Клика, которая не содержится в клике большего размера, называется максимальной по включению.*

Определение 3 *Вершинной раскраской графа G = (V, E) называется такое отображение f: $N \to N$, что $c(v) \neq c(w)v, w$.*

Задача о Максимальной Клике (так же называется MCP) состоит в том, чтобы для заданного графа G найти клику максимального размера.

2 Подходы к решению задачи

2.1 Историческая справка

Было предложено множество решений, в том числе Harary and Ross (предложили решение в 1957), Tarjan and Trojanowski (1977), но с учетом того, что задача NP-полная (NP-complete), наиболее точным на практике часто оказывается эвристический алгоритм Bron and Kerbosch, предложенный в 1973. Но он решает более общую задачу, поэтому до сих пор разрабатываются другие точные алгоритмы для решения MCP, например Li and Quan(MaxCLQ, 2010) или Segundo et al.(BBMCX, 2015).

Задача также сложна для параметрического приближения и трудно аппроксимируема. Но мы рассмотрим несколько алгоритмов для работы с кликами, работающих либо за экспоненциальное время (такие как алгоритм Брона — Кербоша, который я реализую в рамках проекта), и специализирующиеся на семействах графов, таких как планарные графы или совершенные графы, для которых задача может быть решена за полиномиальное время.

2.2 Решение MaxCLQ

Рассмотрим алгоритм MaxCLQ(G, C, LB), основанный на методе ветвей и границ.
На вход подается граф G = (V, E), и клика C, а так же мощность (размер) найденной клики на этом этапе.
Алгоритм отдает C⊔K, K - максимальная клика G, такая что в объединении с C дает мощность $>$ LB, иначе пустое множетсво.

```
1  begin
2  |  if |V|=0 then return C;
3  |  UB ← overestimation(G)+|C|;
4  |  if LB ≥ UB then return ∅;
5  |  select a vertex v from G of the minimum degree;
6  |  C1 ← MaxCLQ(Gv, C∪{v}, LB);
7  |  if |C1| > LB then LB ← |C1| ;
8  |  C2 ← MaxCLQ(G\v, C, LB);
9  |  if |C1| ≥ |C2| then return C1; else return C2;
10 end
```

Это довольно простой алгоритм, при этом достаточно эффективный в реальных жизненных задачах.

2.3 Усовершенствованное решение

Алгоритмы, основанные на методе ветвей и границ для Maxclique часто используют эвристические решения GCP, которые можно получить в разумных пределах. время в качестве их верхней границы на основе следующего свойства: Пусть w(G) обозначает мощность макси- мама клики графа G. Если G можно разбить на k независимых множеств, то w(G) <= k.

В 2002 году появляется алгоритм, который делит G на незавимые множества в качестве препроцессинга. Пока можно добавить вершину в независимое множества, одна из таких вершин (берется вершина с наибольшей степенью) добавляется. Затем конструируется максиимальная клика путем собирания этих же вершин в обратном порядке добавления в независимые множетсва. Фальхе (2002) улучшает алгоритм Каррагана. и Пардалос (1990), используя конструктивную эвристику DSATUR для окраски вершин одну за другой и путем разделения графа на независимые множества.
MCQ (Tomita Seki 2003) раскрашивает вершины в заранее заданном порядке. Предположим, что текущие независимые множества S_1, S_2, ..., S_k (именно в этом порядке, k равно 0 в начале), MCQ вставляет текущую первую вершину v в первый S_i такой, что v не связан со всеми вершинами, уже входящими в S_i. Если такого множетсва не существует, то новое независимое множество S_{k+1} создается и сюда вставляется v. Вершины затем переупорядочиваются в соответствии с их независимым множеством, вершины из S_i предшествуют вершинам из S_j, если i $<$ j. Этот процесс окраски выполняется после каждого ветвления на вершине v.

MCR (Tomita and Kameda, 2007) улучшает MCQ за счет лучшего начального порядка вершин. в исходном входном графе, но использует тот же процесс раскраски чтобы вычислить верхнюю границу.

3 NP-полнота задачи

Мы знаем, что задача о независимом множестве вершин NP-полная(находится в списке Карпа 21 NP-полных задач). 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.

4 Решение задачи

4.1 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.

4.2 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.

4.3 Псевдокод алгоритма

Есть несколько видов алгоритма: with pivoting и without pivoting. Базовая форма алгоритма Брона-Кербоша представляет собой рекурсивный алгоритм поиска с возвратом, который ищет все максимальные клики в заданном графе G. В более общем смысле, по трем непересекающимся наборам вершин R, P и X он находит максимальные клики, включающие все вершин в R, некоторых вершин в P и ни одной вершины в X. При каждом вызове алгоритма P и X представляют собой непересекающиеся множества, объединение которых состоит из тех вершин, которые образуют клики при добавлении к R. В других Другими словами, P⊔X — это набор вершин, которые соединены с каждым элементом R. Когда P и X оба пусты, нет дополнительных элементов, которые можно добавить к R, поэтому R — максимальная клика, и алгоритм выводит R.

```
algorithm BronKerbosch1(R, P, X) is
    if P and X are both empty then
        report R as a maximal clique
    for each vertex v in P do
        BronKerbosch1(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))
        P := P \ {v}
        X := X ⋃ {v}
```

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. Чтобы сэкономить время и позволить алгоритму быстрее возвращаться из ситуаций, которые не содержат максимальных клик, Брон и Кербош представили 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 BronKerbosch2(R, P, X) is
    if P and X are both empty then
        report R as a maximal clique
    choose a pivot vertex u in P ⋃ X
    for each vertex v in P \ N(u) do
        BronKerbosch2(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))
        P := P \ {v}
        X := X ⋃ {v}
```

4.4 Доказательство ассимптотики

Чтобы оценить, как в худшем случае работает алгоритм, докажем утверждение: Утверждение. Через $f(n)$ обозначим количество максимальных клик $n >= 2 =>$

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

Нетрудно проверить 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 $c(G) = j_1 * j_2 * ... * j_t$.

$c(G)$ достигает максимального значения, если максимальное количество непересекающихся множеств содержат 3 вершины, а оставшиеся могут иметь по 2 если остаток 2, а иначе 4, если остаток 1.

Определение 4 *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.*

Согласно результатам, граф с 3n вершинами может содержать максимум 3^n наибольших клики. Графы, удовлетворяющие этой границе — это графы Муна — Мозера $K_{3,3,...}$ — специальный случай графов Турана.

4.5 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 []

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)

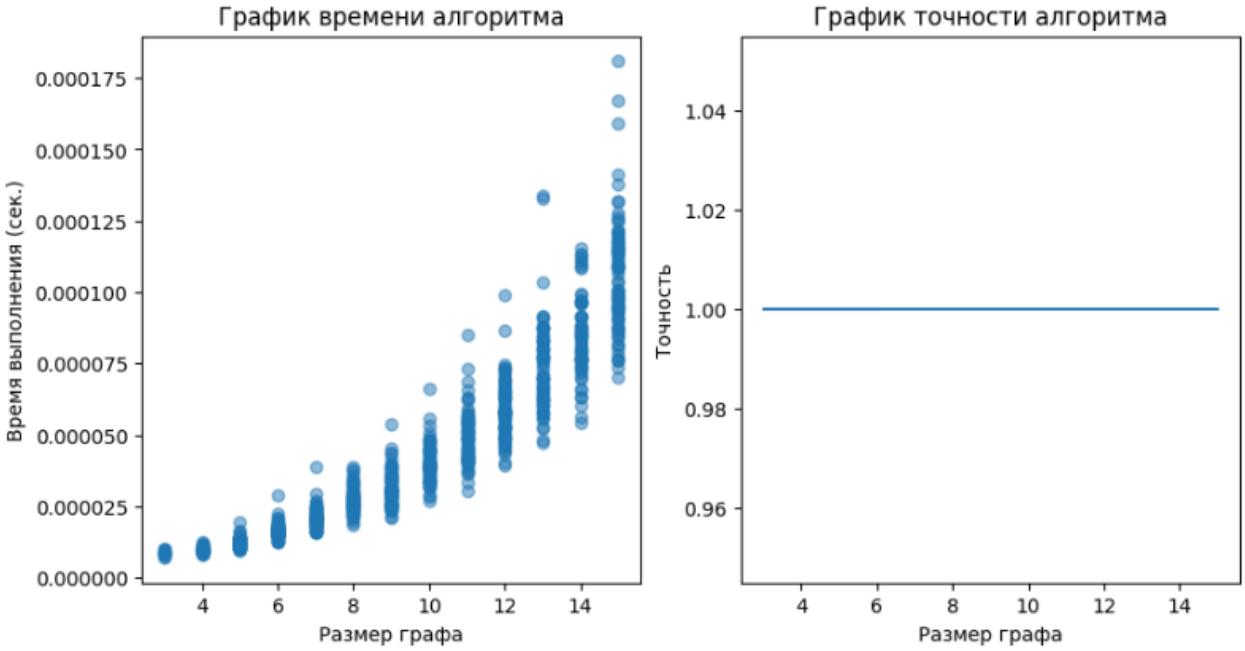
3) Отдельно также написаны тесты для дистанционного и планарного графа

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.

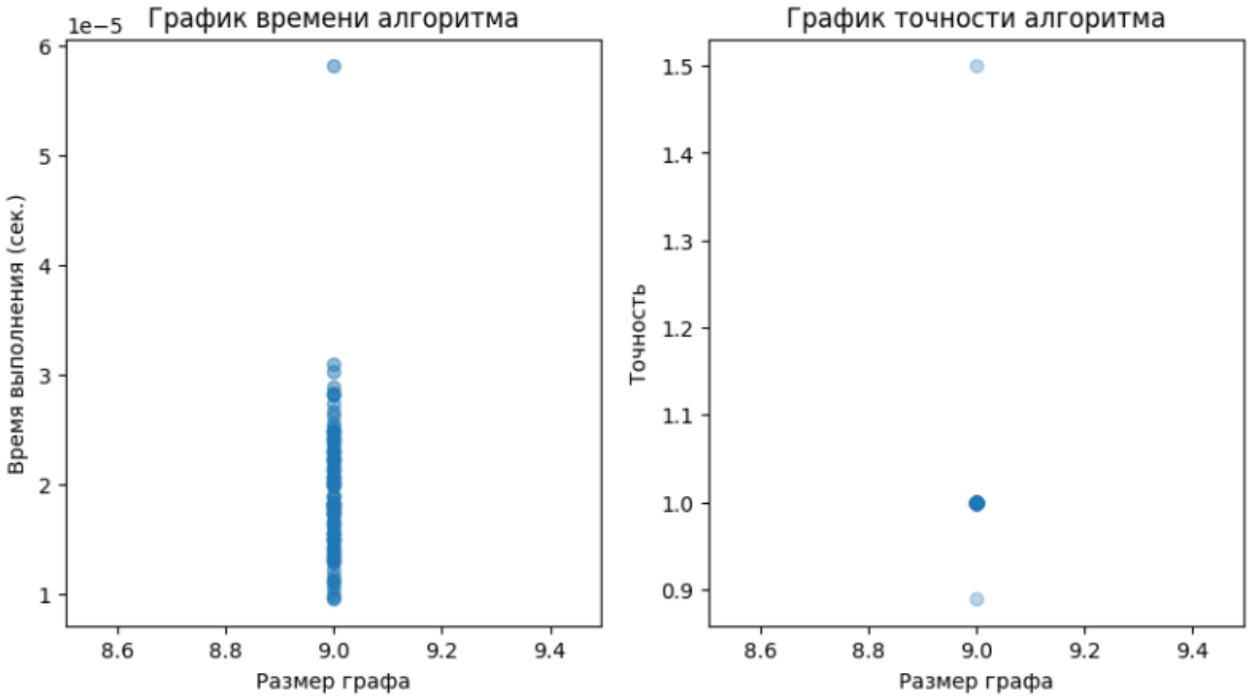
4.6 Простые графы

Для изучения работы алгоритма на простых графах взяли рандомное количество вершин от 3 до 15 и посмотрели на график зависимости времени и точности. Они нам нужны в большом количестве не для сравнения с остальными, а чтобы посмотреть насколько большой разброс по времени и точности получается в общем случае (их недолго и несложно генерировать).

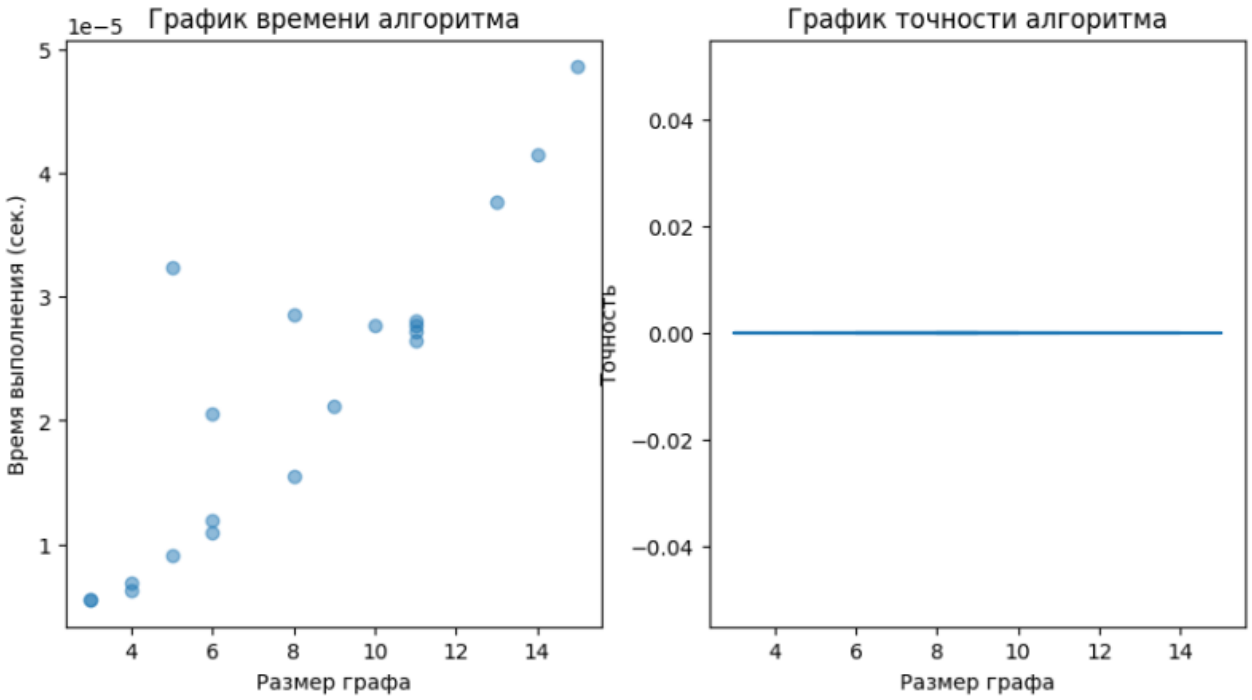


4.7 Графы с фиксированным числом ребер

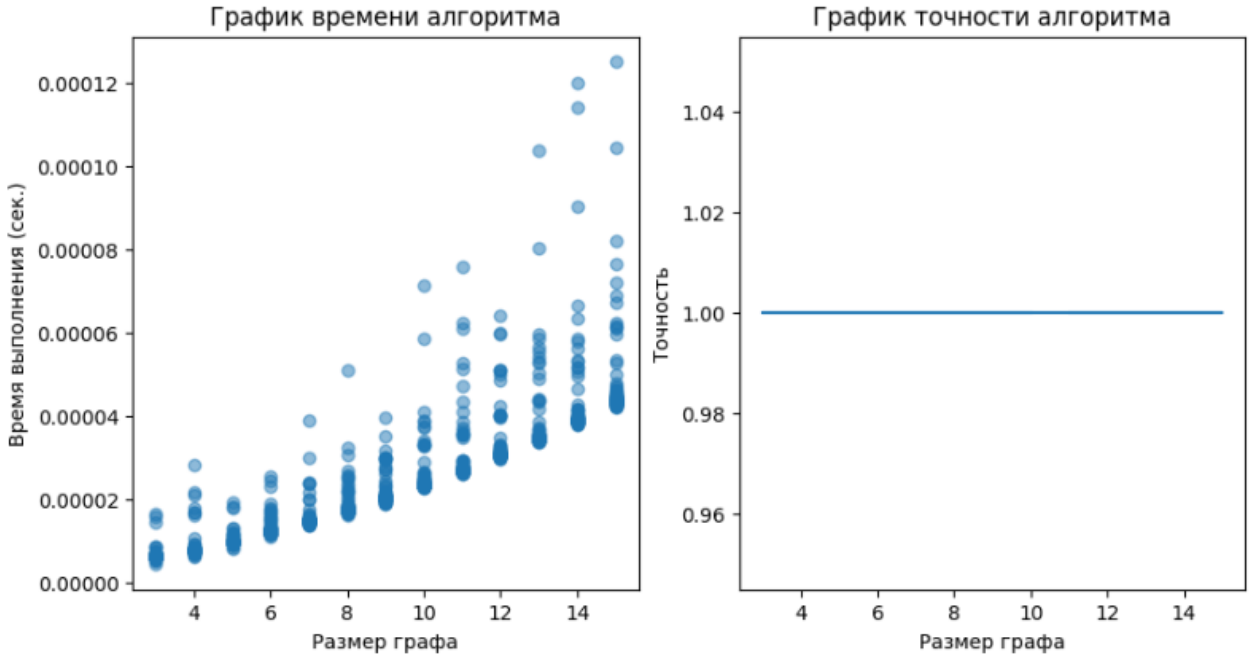
Для изучения работы алгоритма на графах с фиксированной мощностью множетсва ребер взяли рандомное количество 9 вершин и посмотрели на график зависимости времени и точности



4.8 Эйлеровы графы

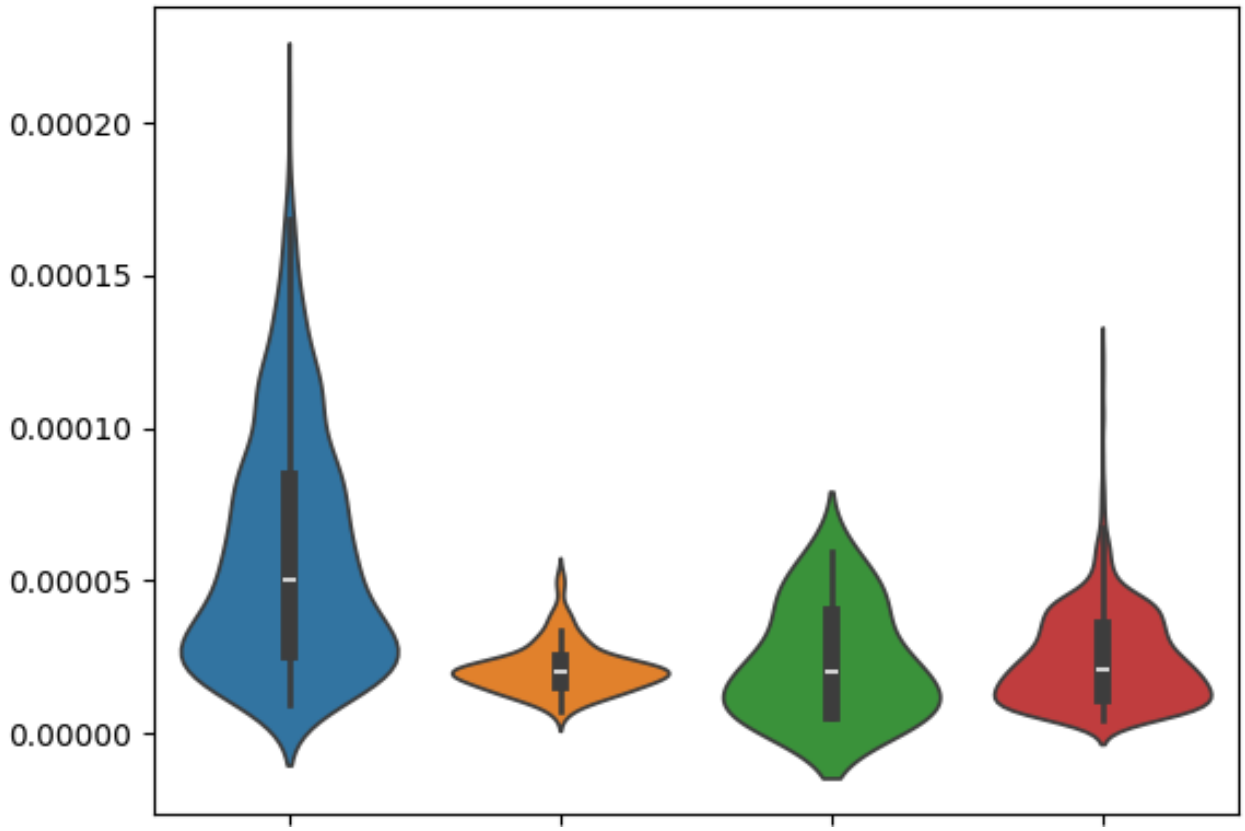


4.9 Полные графы



4.10 Сравнение работы алгоритма на простых, с фиксированным числом ребер, эйлеровых и полных графах

Непростых графов меньше в выборке, чем простых, поэтому вывод отсюда про них сделать не можем (в выборке по простым 1000 штук, остальных по 100 чтобы их можно было сравнить между собой). Однако можем сделать вывод из сравнения графов о том, что на полных графах в общем случае алгоритм работает хуже чем на случайных или каких-то специфических.



5 Вывод

Исходя из графиков, можно сделать вывод, алгоритм хотя и эвристический, но почти всегда работает с точностью около 100%. При этом время работы Так же можем заметить, что лучше всего алгоритм работает на Эйлеровых графах, т.к. на большин- стве тестов точность находится около 100Как 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.
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.

6 Список литературы

[1] Википедия. Задача о клике - Википедия, свободная энциклопедия, 2023. [Онлайн; загружено 1 июня 2024]. URL: https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BA%D0%BB%D0%B8%D0%BA%D0%B5

[2] Википедия. Задача о независимом множестве - Википедия, свободная энциклопедия, 2023. [Онлайн; загружено 1 июня 2024]. URL: https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BD%D0%B5%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D0%BC_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5

[3] Microsoft learn - MSDN Magazine Issues - 2011, Ноябрь. URL: https://learn.microsoft.com/ru-ru/archive/msdn-magazine/2011/november/test-run-greedy-algorithms-and-maximum-clique

[4] ИСА РАН - Вычислительный подход к решению задачи о поиске максимальной клики. М. А. Грибков, А. В. Алексеевский, С. А. Спирин, М. А. Короткова. URL: http://www.isa.ru/proceedings/images/documents/2006-25/185-192.pdf

[5] Википедия. Алгоритм Брона-Кербоша - Википедия, свободная энциклопедия, 2023. [Онлайн; загружено 1 июня 2024]. URL: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D1%80%D0%BE%D0%BD%D0%B0_%E2%80%94_%D0%9A%D0%B5%D1%80%D0%B1%D0%BE%D1%88%D0%B0