ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

**Программа подготовки бакалавров по направлению
01.03.02 Прикладная математика и информатика**

Антонов Илья Витальевич

КУРСОВАЯ РАБОТА

Эффективные линейно-алгебраические алгоритмы поиска заданных подструктур в графах

Научный руководитель

д.ф.-м.н., проф.

Д.С. Малышев

Нижний Новгород, 2023

Содержание

Введение **3**

1 Теория для построения алгоритма **5**

1.1 Обозначения 5

1.2 Терминология из теории графов 7

1.3 Теорема для проверки смежности клик 8

2 Линейно-алгебраический алгоритм перечисления и подсчета k-клик **10**

2.1 Идея из статьи Нешетрила-Поляка 10

2.2 Редукция матрицы смежности клик 10

2.3 Линейно-алгебраический алгоритм перечисления k-клик 12

2.4 Линейно-алгебраический алгоритм подсчета k-клик 14

2.5 Замечание о редукции 16

3 Стандарт GraphBLAS **18**

4 Имплементация алгоритма **20**

4.1 Проверка корректности алгоритма 20

4.2 Вычислительные эксперименты 22

4.2.1 Набор данных ca-CondMat 23

4.2.2 Набор данных ca-HepTh 23

4.2.3 Набор данных com-DBLP 24

4.2.4 Набор данных com-Amazon 24

4.2.5 Набор данных com-Youtube 25

Заключение **26**

Список использованной литературы **28**

Введение

Задача подсчета количества подграфов, имеющих определенную структуру, встречается во многих приложениях теории графов. Частным её случаем является задача подсчета k-клик, которая представляет особый интерес поскольку любую задачу поиска подструктуры в графе можно свести к задаче поиска k-клики [13]. Известно множество алгоритмов подсчета k-клик [1], [2], [4], [6], [9], [10], [12], [13], [14], [15], [16]. Часть из перечисленных алгоритмов являются параллельными. Подобная необходимость объясняется тем, что в приложениях, например при анализе социальных сетей, возникают графы больших размерностей. Их возможно обработать за разумное время, лишь применяя суперкомпьютерные технологии, которые требуют от алгоритма прежде всего хорошую шкалируемость.

С недавних пор набирают популярность алгоритмы на графах в терминах линейно-алгебраических операций над матрицами смежности и инцидентности. На настоящий момент науке известно большое количество паралелльных алгоритмов для умножения матриц, что делает язык линейной алгебры подходящим средством для построения параллельных алгоритмов на графах. Актуальность такого подхода подтверждается выходом книги Кепнера и Гильберта [8], а также появлением стандарта GraphBLAS, развитие которого спонсируют такие компании как IBM, Intel, Nvidia и прочие.

До недавних пор было известно лишь небольшое количество линейно - алгебраических алгоритмов подсчета k-клик. Все они решали задачу лишь в частном случае. Так, например, при $k = 3$ предлагался алгоритм в работе [13], а при $k = 4$ в работе [5]. Однако ситуация улучшилась с выходом работы Емелина, Хлюстова, Малышева, Развенской [3], которая основывается на общей концепции алгоритма перечисления k-клик из статьи Нешетрила-Поляка [13] и обобщает его в терминах линейно-алгебраических операций.

Алгоритм подсчета k-клик в настоящей курсовой работе основывается на идее Нешетрила-Поляка [13] и на алгоритме из работы [3], однако имеются некоторые различия с последней работой. Так, например, то, что в терминах данной курсовой работы называется редукцией матрицы смежности клик, по концепции похоже на то, что в работе [3] называется ациклическим ориенти-

рованием графа клик. Но для этого в настоящей курсовой работе используется другой алгоритм. Также в данной работе линейно-алгебраический алгоритм подсчета k-клик умножает матрицы смежности клик и не строит матрицу инцидентности k-клик вершинам графа. Еще одной отличительной особенностью является то, что весь псевдокод написан только в линейно-алгебраических терминах и не требует дополнительных преобразований при реализации.

1 Теория для построения алгоритма

1.1 Обозначения

При работе с матрицами и векторами будем соблюдать следующие соглашения:

1. $\boldsymbol{A}(i, j)$ - элемент матрицы $\boldsymbol{A}$, находящийся в i-ой строке j-го столбца
2. $\boldsymbol{A}(i, :)$ - i-я строка матрицы $\boldsymbol{A}$
3. $\boldsymbol{A}(:, j)$ - j-й столбец матрицы $\boldsymbol{A}$
4. $\boldsymbol{A}^T$ - матрица транспонированная к матрице $\boldsymbol{A}$
5. $\boldsymbol{AB}$ - скалярное умножение матриц $\boldsymbol{A}$ и $\boldsymbol{B}$ в полукольце с стандартными операциями сложения и умножения
6. $\boldsymbol{A}^2 = \boldsymbol{AA}$
7. $\boldsymbol{A} \max . * \boldsymbol{B}$ - скалярное умножение матриц $\boldsymbol{A}$ и $\boldsymbol{B}$ в полукольце, где операция сложения заменена операцией взятия максимума, а операция умножения стандартная
8. $\boldsymbol{A} \min . * \boldsymbol{B}$ - скалярное умножение матриц $\boldsymbol{A}$ и $\boldsymbol{B}$ в полукольце, где операция сложения заменена операцией взятия минимума, а операция умножения стандартная
9. $\boldsymbol{A}$ any $. < \boldsymbol{B}$ - скалярное умножение матриц $\boldsymbol{A}$ и $\boldsymbol{B}$ в полукольце, где операция сложения заменена на логическое или, а операций умножения проверяет высказывание $x < y$
10. $\boldsymbol{A} + \boldsymbol{B}$ - поэлементное матричное сложение с стандартной операцией сложения
11. $\boldsymbol{A} * \boldsymbol{B}$ - поэлементное матричное умножение с стандартной операцией умножения

12. $[\boldsymbol{A} = x]$ - матрица, заданная по правилу:

$$[\boldsymbol{A} = x](i, j) = \begin{cases} 1, & \boldsymbol{A}(i, j) = x \\ 0, & \boldsymbol{A}(i, j) \neq x \end{cases}$$

13. Пусть $\boldsymbol{A}$ - матрица размера $n \times m$, и задан индексный вектор $I = (i_1, \ldots, i_p)$, где $i_k \in \{1, 2, \ldots, n\}$ при $k = \overline{1, p}$. Тогда $\boldsymbol{A}(I, :)$ - матрица размера $p \times m$, заданная по правилу:

$$\boldsymbol{A}(I, :)(k, j) = \boldsymbol{A}(i_k, j)$$

14. $\boldsymbol{I}_n$ - единичная матрица размера $n \times n$

15. $\mathrm{triu}(\boldsymbol{A})$ - верхнетреугольная матрица, определенная по правилу:

$$\mathrm{triu}(\boldsymbol{A})(i, j) = \begin{cases} \boldsymbol{A}(i, j), & \text{если } i > j \\ 0, & \text{иначе} \end{cases}$$

16. $\mathrm{sum}(\boldsymbol{A})$ - сумма элементов матрицы $\boldsymbol{A}$

17. $\dim(\boldsymbol{v})$ - размерность вектора $\boldsymbol{v}$

18. 0_n - нулевой вектор размера n

19. $\boldsymbol{v}(i)$ - i-й элемент вектора $\boldsymbol{v}$

20. $\mathrm{nnz}(\boldsymbol{v}) = \{i : \boldsymbol{v}(i) \neq 0\}$ - мн-во индексов ненулевых элементов вектора $\boldsymbol{v}$

Также приведем прочие обозначения:

1. $|\boldsymbol{M}|$ - мощность множества $\boldsymbol{M}$

2. $x \operatorname{div} y$ - целая часть от деления x на y

3. $x \bmod y$ - остаток от деления x на y

4. $\binom{n}{n_1, n_2, \ldots, n_k} = \dfrac{n!}{n_1! n_2! \ldots n_k!}$ - мультиномиальный коэффициент, где $n = n_1 + n_2 + \ldots + n_k$

1.2 Терминология из теории графов

В данной работе рассматриваются простые (без петель и кратных ребер) неориентированные графы $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ с множеством вершин $\boldsymbol{V}$ и множеством ребер $\boldsymbol{E}$.

Приведем определения и обозначения, которые будут использоваться в дальнейшем при работе с графами. Положим для лаконичности записи $|\boldsymbol{V}| = n$.

Определение 1.1. *Матрицей смежности называется матрица* $\boldsymbol{A_G} \in \{0,1\}^{n \times n}$, *которая задается по правилу:*

$$\boldsymbol{A_G}(i,j) = \begin{cases} 1, & (i,j) \in \boldsymbol{E} \\ 0, & (i,j) \notin \boldsymbol{E} \end{cases}$$

Множество вершин, смежных с вершиной $i \in \boldsymbol{V}$ обозначим как $Neighbors(i)$.

Определение 1.2. *Кликой называется подмножество множества вершин графа такое, что любая пара вершин в нем смежна. Если мощность клики равна* k, *то такую клику будем назвать* k*-кликой или кликой размера* k.

Для удобства клики будем помечать и обозначать клику с номером i как $Clique(i)$.

Определение 1.3. *Пусть выбран набор клик* $\{Clique(i)\}_{i=1}^{p}$. *Матрицей инцидентности клик вершинам графа называется матрица* $\boldsymbol{C} \in \{0,1\}^{p \times n}$, *построенная по правилу:*

$$\boldsymbol{C}(i,j) = \begin{cases} 1, & j \in Clique(i) \\ 0, & j \notin Clique(i) \end{cases}$$

Другими словами, i-я строка матрицы $\boldsymbol{C}$ кодирует клику $Clique(i)$ и содержит единицы на тех позициях, которые соответствуют вершинам, находящимся в рассматриваемой клике.

Определение 1.4. *Клики* $Clique(i)$ *и* $Clique(j)$ *называются смежными, если любая вершина из клики* $Clique(i)$ *смежна с любой вершиной из клики* $Clique(j)$.

Заметим, что по такому определению если клики $Clique(i)$ и $Clique(j)$ смежны, то $Clique(i) \cap Clique(j) = \emptyset$.

Покажем это от противного. Пусть $x \in Clique(i) \cap Clique(j)$. Тогда в силу смежности клик $Clique(i)$ и $Clique(j)$, вершина x должна быть смежна сама с собой. Получаем противоречие с тем, что граф простой (без петель).

Определение 1.5. *Пусть даны два набора клик* $\{Clique_1(i)\}_{i=1}^{p_1}$, $\{Clique_2(j)\}_{j=1}^{p_2}$. *Матрицей смежности первого и второго наборов клик называется матрица* $\boldsymbol{B} \in \{0,1\}^{p_1 \times p_2}$, *которая задается по правилу:*

$$\boldsymbol{B}(i,j) = \begin{cases} 1, \ Clique_1(i) \text{ и } Clique_2(j) \text{ смежны} \\ 0, \ \text{иначе} \end{cases}$$

1.3 Теорема для проверки смежности клик

В данном разделе будет установлена теорема, которая позволит проверять смежность клик в терминах линейно-алгебраических операций. Для начала изложим следующую лемму:

Лемма 1.6. *Пусть дан граф* $\boldsymbol{G}$ *и его матрица смежности* $\boldsymbol{A_G} \in \{0,1\}^{n \times n}$. *В этом графе выбран набор клик* $\{Clique(i)\}_{i=1}^{p}$, *по которому построена матрица инцидентности клик вершинам графа* $\boldsymbol{C} \in \{0,1\}^{p \times n}$. *Тогда:*

$$|Clique(i) \cap Neighbors(j)| = l \Leftrightarrow \boldsymbol{[CA_G]}(i,j) = l$$

Доказательство. 1. Пусть $|Clique(i) \cap Neighbors(j)| = l$. Введем обозначение $\boldsymbol{M} = Clique(i) \cap Neighbors(j)$. Поскольку вектор-строка $\boldsymbol{C}(i,:)$ кодирует клику $Clique(i)$, вектор-столбец $\boldsymbol{A_G}(:,j)$ кодирует соседей вершины j, и при любом $k = \overline{1,n}$ элементы $\boldsymbol{C}(i,k), \boldsymbol{A_G}(k,j) \in \{0,1\}$, то имеем:

$$\boldsymbol{[CA_G]}(i,j) = \boldsymbol{C}(i,:)\boldsymbol{A_G}(:,j) = \sum_{k=1}^{n} \boldsymbol{C}(i,k)\boldsymbol{A_G}(k,j) = \sum_{k \in \boldsymbol{M}} 1 = l$$

2. Пусть $\boldsymbol{[CA_G]}(i,j) = l$. Известно, что

$$\boldsymbol{[CA_G]}(i,j) = \boldsymbol{C}(i,:)\boldsymbol{A_G}(:,j) = \sum_{k=1}^{n} \boldsymbol{C}(i,k)\boldsymbol{A_G}(k,j)$$

Так как $\boldsymbol{C}(i,k)$, $\boldsymbol{A_G}(k,j) \in \{0,1\}$, то из определения матричного умножения следует, что в векторах $\boldsymbol{C}(i,:)$ и $\boldsymbol{A_G}(:,j)$ существует ровно l позиций на которых одновременно стоят 1, и $n-l$ позиций на которых хотя бы в одном векторе стоит 0. Из этого непосредственно следует, что $|Clique(i) \cap Neighbors(j)| = l$. □

Теорема 1.7. *Пусть дан граф* $\boldsymbol{G}$ *и его матрица смежности* $\boldsymbol{A_G} \in \{0,1\}^{n\times n}$. *В этом графе выбраны два набора клик* $\{Clique_1(i)\}_{i=1}^{p_1}$ *и* $\{Clique_2(j)\}_{j=1}^{p_2}$, *по которым построены матрицы инцидентности клик вершинам графа* $\boldsymbol{C}_1 \in \{0,1\}^{p_1\times n}$ *и* $\boldsymbol{C}_2 \in \{0,1\}^{p_2\times n}$ *соответственно. Тогда:*

$$Clique_1(i) \text{ и } Clique_2(j) \text{ смежны} \Leftrightarrow [\boldsymbol{C}_1\boldsymbol{A}\boldsymbol{C}_2^{\boldsymbol{T}}](i,j) = |Clique_1(i)| \cdot |Clique_2(j)|$$

Доказательство. 1. Пусть клики $Clique_1(i)$ и $Clique_2(j)$ смежны. Значит для любого $k \in Clique_2(j)$ выполнено $|Clique_1(i) \cap Neighbors(k)| = |Clique_1(i)|$, и применяя лемму 1.6 получаем $[\boldsymbol{C}_1\boldsymbol{A_G}](i,k) = |Clique_1(i)|$. Также заметим, что $\boldsymbol{C}_2^{\boldsymbol{T}}(:,j)$ это вектор-столбец, который кодирует клику $Clique_2(j)$. Тогда:

$$[\boldsymbol{C}_1\boldsymbol{A_G}\boldsymbol{C}_2^{\boldsymbol{T}}](i,j) = [\boldsymbol{C}_1\boldsymbol{A_G}](i,:)\boldsymbol{C}_2^{\boldsymbol{T}}(:,j) = \sum_{k=1}^{n} [\boldsymbol{C}_1\boldsymbol{A_G}](i,k)\boldsymbol{C}_2^{\boldsymbol{T}}(k,j) =$$

$$= \sum_{k\in Clique_2(j)} [\boldsymbol{C}_1\boldsymbol{A_G}](i,k) = \sum_{k\in Clique_2(j)} |Clique_1(i)| = |Clique_1(i)| \cdot |Clique_2(j)|$$

2. Пусть $[\boldsymbol{C}_1\boldsymbol{A_G}\boldsymbol{C}_2^{\boldsymbol{T}}](i,j) = |Clique_1(i)| \cdot |Clique_2(j)|$. Выше показано, что:

$$[\boldsymbol{C}_1\boldsymbol{A_G}\boldsymbol{C}_2^{\boldsymbol{T}}](i,j) = \sum_{k\in Clique_2(j)} [\boldsymbol{C}_1\boldsymbol{A_G}](i,k)$$

Эта сумма состоит из $|Clique_2(j)|$ слагаемых и при любом $k \in Clique_2(j)$ выполнено неравенство $0 \le [\boldsymbol{C}_1\boldsymbol{A_G}](i,k) \le |Clique_1(i)|$. Она может равняться $|Clique_1(i)|\cdot|Clique_2(j)|$ только, если каждое её слагаемое принимает своё максимально возможное значение $|Clique_1(i)|$. Значит при $k \in Clique_2(j)$ выполнено $[\boldsymbol{C}_1\boldsymbol{A_G}](i,k) = |Clique_1(i)|$. Поэтому по лемме 1.6:

$$|Clique_1(i) \cap Neighbors(k)| = |Clique_1(i)|$$

Другими словами k смежна со всеми вершинами из $Clique_1(i)$. Применяя данную процедуру для всех $k \in Clique_2(j)$ устанавливаем, что клики $Clique_1(i)$ и $Clique_2(j)$ смежны. □

2 Линейно-алгебраический алгоритм перечисления и подсчета k-клик

Далее будем предполагать, что задан простой (без петель и кратных ребер) неориентированный граф $\boldsymbol{G}$ с матрицей смежности $\boldsymbol{A_G} \in \{0,1\}^{n\times n}$.

2.1 Идея из статьи Нешетрила-Поляка

За основу для линейно-алгебраического алгоритма подсчета k-клик будет взята идея из статьи Нешетрила-Поляка [13]. В данной работе приводится методология перечисления/подсчета треугольников в графе, то есть клик размера 3. Предлагается вычислить матрицу $\boldsymbol{A_G^2}$, элемент $\boldsymbol{A_G^2}(i,j)$ которой это количество путей длины 2 между вершинами i и j. Затем утверждается: если $0 < \boldsymbol{A_G}(i,j) \leq \boldsymbol{A_G^2}(i,j)$, то вершины i и j лежат в некоторых треугольниках. Количество таких треугольников равно $\boldsymbol{A_G^2}(i,j)$, и вершины, замыкающие треугольник, лежат в множестве $Neighbors(i) \cap Neighbors(j)$.

Идею перечисления/подсчета треугольников можно обобщить на случай перечисления/подсчета $3l$-клик. Для этого строится граф, вершинами в котором являются l-клики, а ребра между вершинами проводятся тогда и только тогда, когда задающие вершины l-клики смежны в смысле определения 1.4. Затем в построенном графе применяется изложенный выше алгоритм перечисления/подсчета треугольников. Если в таком графе обнаружить треугольник, то в исходном графе будет найдена $3l$-клика.

Также приводится следующее соображение. Если найдены $3l$-клики, то можно обнаружить клики размеров $3l + 1$ и $3l + 2$. Для этого достаточно к $3l$-клике добавить 1 или 2 вершины, смежные со всеми вершинами из $3l$-клики соответственно.

2.2 Редукция матрицы смежности клик

Для хранения l-клик будем использовать матрицу инцидентности l-клик вершинам графа $\boldsymbol{C_l}$. Используя теорему 1.7, можно найти матрицу смежности l и l клик $\boldsymbol{M_{l,l}} = \boldsymbol{[C_l A_G C_l^T = l^2]}$. Однако данная матрица обладает существенным недостатком, а именно при использовании определения 1.4 смежности клик,

клика размера $3l$ будет выражаться через клики размера l, l, l не единственным образом. Исправим этот недостаток, усилив определение смежности клик.

Определение 2.1. *Клики* $\boldsymbol{C}_l(i,:)$ *и* $\boldsymbol{C}_l(j,:)$ *смежны тогда и только тогда, когда они смежны по определению 1.4 и* $\max(nnz(\boldsymbol{C}_l(i,:))) < \min(nnz(\boldsymbol{C}_l(j,:)))$.

По такому определению смежности граф l-клик будет ориентированным и ациклическим.

Для того чтобы в линейно-алгебраических терминах вычислить матрицу смежности с усиленным определением смежности необходимы векторы:

$$\boldsymbol{w}_{max} = \boldsymbol{C}_l \max .* (1, 2, \ldots, n)^T$$

$$\boldsymbol{w}_{min} = \boldsymbol{C}_l \min .* (1, 2, \ldots, n)^T$$

По ним вычисляется битовая матрица-маска:

$$\boldsymbol{W} = \boldsymbol{w}_{max} \text{ any } . < \boldsymbol{w}_{min}^T$$

Наложим эту матрицу-маску на $\boldsymbol{M}_{l,l}$ и получим новую матрицу смежности l и l клик:

$$\boldsymbol{M}_{l,l} = \boldsymbol{M}_{l,l} * W$$

Аналогичным способом можно вычислить матрицу смежности l и r клик в новых терминах смежности. Приведем алгоритм получения матрицы смежности клик в общем случае.

Алгоритм 1 Получение матрицы смежности l и r клик

1: **function** get_adjacency_matrix($\boldsymbol{A_G}, \boldsymbol{C}_l, \boldsymbol{C}_r$)
2: $\quad \boldsymbol{M}_{l,r} \leftarrow [\boldsymbol{C_l A_G C_r^T = l \cdot r}]$
3: $\quad \boldsymbol{w}_{max} \leftarrow \boldsymbol{C}_l \max .* (1, 2, \ldots, n)^T$
4: $\quad \boldsymbol{w}_{min} \leftarrow \boldsymbol{C}_r \min .* (1, 2, \ldots, n)^T$
5: $\quad \boldsymbol{W} \leftarrow \boldsymbol{w}_{max}\ any. < \boldsymbol{w}_{min}^T$
6: $\quad \boldsymbol{M}_{l,r} \leftarrow \boldsymbol{M}_{l,r} * W$
7: $\quad$ **return** $\boldsymbol{M}_{l,r}$
8: **end function**

2.3 Линейно-алгебраический алгоритм перечисления k-клик

Приведем линейно-алгебраический алгоритм перечисления k-клик. Он требует матрицу смежности графа $\boldsymbol{A_G}$ и результатом его работы является найденная матрица инцидентности k-клик вершинам графа $\boldsymbol{C}_k$.

Пусть $l = k \operatorname{div} 3$ и $r = k \bmod 3$. Для построения матрицы инцидентности k-клик вершинам графа алгоритм рекурсивно запрашивает у себя же матрицу инцидентности l-клик вершинам графа $\boldsymbol{C}_l$. Далее будем считать что матрица $\boldsymbol{C}_l$ дана.

Используя алгоритм 1, получим матрицу смежности l и l клик $\boldsymbol{M}_{l,l}$. Затем вычислим матрицу $\boldsymbol{B} = \boldsymbol{M}_{l,l} * \boldsymbol{M}_{l,l}^2$. Если $\boldsymbol{B}(i,j) \neq 0$, то клики $\boldsymbol{C_l}(i,:)$ и $\boldsymbol{C_l}(j,:)$ образуют ровно $\boldsymbol{B}(i,j)$ треугольников. Получим из матрицы $\boldsymbol{B}$ два индексных вектора $\boldsymbol{I_B}$ и $\boldsymbol{J_B}$, которые содержат индексы её ненулевых элементов.

Вычислим матрицу $\boldsymbol{S} = \boldsymbol{M}_{l,l}(\boldsymbol{I_B},:) * \boldsymbol{M}_{l,l}^T(\boldsymbol{J_B},:)$. Она обладает следующим свойством. Если $\boldsymbol{S}(i,k) = 1$, то $\boldsymbol{C}_l(\boldsymbol{I_B}(i),:) + \boldsymbol{C}_l(\boldsymbol{J_B}(i),:) + \boldsymbol{C}_l(k,:)$ - $3l$-клика. Используя вышеприведенное свойство, получим 3 вектора I, J, K, такие что $3l$-кликой является $\boldsymbol{C}_l(I(i),:) + \boldsymbol{C}_l(J(i),:) + \boldsymbol{C}_l(K(i),:)$. По этим векторам составим матрицу инцидентности $3l$-клик вершинам графа:

$$\boldsymbol{C_{3l}} = \boldsymbol{C_l}(I,:) + \boldsymbol{C_l}(J,:) + \boldsymbol{C_l}(K,:)$$

Если $r = 0$, то искомая матрица найдена. Если $r \neq 0$, то используя алгоритм 1, построим матрицу $\boldsymbol{M}_{3l,1}$ а затем извлечем из неё векторы индексов ненулевых элементов $\boldsymbol{I}_{\boldsymbol{M}_{3l,1}}$ и $\boldsymbol{J}_{\boldsymbol{M}_{3l,1}}$.

Матрица $\boldsymbol{C}_{3l+1}$ выражается как $\boldsymbol{C}_{3l}(\boldsymbol{I}_{\boldsymbol{M}_{3l,1}},:) + \boldsymbol{C}_1(\boldsymbol{J}_{\boldsymbol{M}_{3l,1}},:)$. В случае $r = 1$ искомая матрица найдена, а в случае $r = 2$ необходимо повторить этот же шаг уже для матрицы $\boldsymbol{C}_{3l+1}$.

Приведем псевдокод вышеперечисленных алгоритмов. Ниже get_indexes(·) это функция, возвращающая два вектора, которые кодируют индексы ненулевых элементы переданной матрицы.

Алгоритм 2 Перечисление $l+1$-клик

function increment_cliques($\boldsymbol{A_G}, \boldsymbol{C}_l$)
    $\boldsymbol{C}_1 \leftarrow \boldsymbol{I}_n$
    $\boldsymbol{M}_{l,1} \leftarrow$ get_adjacency_matrix($\boldsymbol{A_G}, \boldsymbol{C}_l, \boldsymbol{C}_1$)
    $I_{\boldsymbol{M}_{l,1}}, J_{\boldsymbol{M}_{l,1}} \leftarrow$ get_indexes($\boldsymbol{M}_{l,1}$)
    $\boldsymbol{C}_{l+1} \leftarrow \boldsymbol{C}_l(I_{\boldsymbol{M}_{l,1}}, :) + \boldsymbol{C}_1(J_{\boldsymbol{M}_{l,1}}, :)$
    return $\boldsymbol{C}_{l+1}$
end function

Алгоритм 3 Сопоставление индексов

function index_matching($I_{\boldsymbol{S}}$, $I_{\boldsymbol{B}}$, $J_{\boldsymbol{B}}$)
    $n \leftarrow \dim(I_{\boldsymbol{S}})$
    $I \leftarrow 0_n$
    $J \leftarrow 0_n$
    for $i \leftarrow \overline{1,n}$ **do**
        $I(i) \leftarrow I_{\boldsymbol{B}}(I_{\boldsymbol{S}}(i))$
        $J(i) \leftarrow J_{\boldsymbol{B}}(I_{\boldsymbol{S}}(i))$
    end for
    return I, J
end function

Алгоритм 4 Перечисление k-клик

function matching_k_cliques($\boldsymbol{A_G}, k$)
    if $k = 1$ **then**
        return $\boldsymbol{I}_n$
    end if
    if $k = 2$ **then**
        $\boldsymbol{C}_1 \leftarrow$ matching_k_cliques($\boldsymbol{A_G}, 1$)
        $I_{\boldsymbol{A_G}}, J_{\boldsymbol{A_G}} \leftarrow$ get_indexes(triu($\boldsymbol{A_G}$))
        return $\boldsymbol{C}_1(I_{\boldsymbol{A_G}}, :) + \boldsymbol{C}_1(J_{\boldsymbol{A_G}}, :)$
    end if
    $l \leftarrow k \operatorname{div} 3$
    $r \leftarrow k \bmod 3$
    $\boldsymbol{C_l} \leftarrow$ matching_k_cliques($\boldsymbol{A_G}, l$)

Алгоритм 5 Перечисление k-клик (продолжение)

```
    M_{l,l} ← get_adjacency_matrix(A_G, C_l, C_l)
    B ← M_{l,l} * M_{l,l}^2
    I_B, J_B ← get_indexes(B)
    S ← M_{l,l}(I_B, :) * M_{l,l}^T(J_B, :)
    I_S, J_S ← get_indexes(S)
    I, J ← index_matching(I_S, I_B, J_B)
    K ← J_S
    C_{3l} ← C_l(I, :) + C_l(J, :) + C_l(K, :)
    if r = 0 then
        return C_{3l}
    end if
    C_{3l+1} ← increment_cliques(A_G, C_{3l})
    if r = 1 then
        return C_{3l+1}
    end if
    C_{3l+2} ← increment_cliques(A_G, C_{3l+1})
    return C_{3l+2}
end function
```

2.4 Линейно-алгебраический алгоритм подсчета k-клик

Для подсчета k-клик нам потребуется матрица смежности графа $\boldsymbol{A}_G$ и матрица инцидентности l-клик вершинам графа $\boldsymbol{C}_l$. Пусть как и прежде $l = k \text{ div } 3$, $r = k \text{ div } 3$.

В случае $r = 0$ клика размера $3l$ будет выражаться как объединения клик размеров l, l, l. Используя алгоритм 1, получим матрицу смежности l и l клик $\boldsymbol{M}_{l,l}$. Тогда количество k-клик представится как

$$sum(\boldsymbol{M}_{l,l} * \boldsymbol{M}_{l,l}^2)$$

В случае $r = 1$ клика размера $3l + 1$ будет выражаться как объединения клик размеров $l + 1$, l, l. Получим матрицу смежности $l + 1$ и l клик $\boldsymbol{M}_{l+1,l}$, а также l

и l клик $\boldsymbol{M}_{l,l}$. Количество k-клик представится как

$$sum(\boldsymbol{M}_{l+1,l} * (\boldsymbol{M}_{l+1,l}\boldsymbol{M}_{l,l}))$$

В случае $r = 2$ клика размера $3l + 2$ будет выражаться как объединения клик размеров l, $l+1$, $l+1$. Получим матрицу смежности l и $l+1$ клик $\boldsymbol{M}_{l,l+1}$, а также $l+1$ и $l+1$ клик $\boldsymbol{M}_{l+1,l+1}$. Количество k-клик представится как

$$sum(\boldsymbol{M}_{l,l+1} * (\boldsymbol{M}_{l,l+1}\boldsymbol{M}_{l+1,l+1}))$$

Приведем псевдокод данного алгоритма.

Алгоритм 6 Подсчет k-клик

```
1:  function counting_k_cliques(A_G, k)
2:      if k = 1 then
3:          return n
4:      end if
5:      if k = 2 then
6:          return sum(A)/2
7:      end if
8:      l ← k div 3
9:      r ← k mod 3
10:     C_l ← matching_k_cliques(A_G, l)
11:     if r = 0 then
12:         M_{l,l} ← get_adjacency_matrix(A_G, C_l, C_l)
13:         return sum(M_{l,l} * M_{l,l}^2)
14:     end if
15:     if r = 1 then
16:         M_{l+1,l} ← get_adjacency_matrix(A_G, C_{l+1}, C_l)
17:         M_{l,l} ← get_adjacency_matrix(A_G, C_l, C_l)
18:         return sum(M_{l+1,l} * (M_{l+1,l} M_{l,l}))
19:     end if
```

Алгоритм 7 Подсчет k-клик (продолжение)

```
    if r = 2 then
        M_{l,l+1} ← get_adjacency_matrix(A_G, C_l, C_{l+1})
        M_{l+1,l+1} ← get_adjacency_matrix(A_G, C_{l+1}, C_{l+1})
        return sum(M_{l,l+1} * (M_{l,l+1} M_{l+1,l+1}))
    end if
end function
```

2.5 Замечание о редукции

Одним из недостатков редукции матрицы смежности, изложенной в разделе 2.2, является использование в её алгоритме сравнения целых чисел в диапазоне $1, 2, ..., n$, где n - количество в вершин в графе. Если значение n большое (например 100.000), то сравнение целых чисел будет работать медленно, а значит и алгоритм покажет плохую производительность. Поскольку линейно-алгебраический алгоритм подсчета k-клик создается преимущественно для больших графов, то эту проблему нужно каким-либо образом разрешить. Предложим возможный способ.

Для этого предлагается при подсчете количества клик не редуцировать матрицы смежности, а дальше использовать алгоритм 7 без изменений. Однако возникнет следующая проблема. Клика размера $3l$ подсчитается $\binom{3l}{l,l,l}$ раз, клика размера $3l+1$ подсчитается $\binom{3l+1}{l+1,l,l}$ раз, а клика размера $3l+2$ подсчитается $\binom{3l+2}{l+1,l+1,l}$ раза. Приведем таблицу значений этих коэффициентов.

k	3	4	5	6	7	8	9	10	11	12
Вклад клики в сумму	6	12	30	90	210	560	1680	4200	11550	34650

При реализации алгоритма нужно понять какой вклад клики в сумму допустим. Видно, что при $k \leq 6$ этот вклад несущественный, а при $k = 12$ особенно на больших графах может случиться переполнение целочисленного типа при подсчете ответа. Также стоит учесть, что при больших k скорее всего будет эффективнее произвести редукцию, поскольку лишние подсчеты тоже сказываются на производительности алгоритма.

Также предложим алгоритм, который позволит перечислять 3-клики без редукции.

Алгоритм 8 Перечисление 3-клик

function matching_3_cliques($\boldsymbol{A_G}, \boldsymbol{C}_1$)
    $\boldsymbol{B} \leftarrow \boldsymbol{A_G} * \boldsymbol{A_G^2}$
    $\boldsymbol{B} \leftarrow \text{triu}(\boldsymbol{B})$
    $\boldsymbol{I_B}, \boldsymbol{J_B} \leftarrow \text{get\_indexes}(\boldsymbol{B})$
    $\boldsymbol{A_G} \leftarrow \text{triu}(\boldsymbol{A_G})$
    $\boldsymbol{S} \leftarrow \boldsymbol{A_G}(\boldsymbol{I_B}, :) * \boldsymbol{A_G}(\boldsymbol{J_B}, :)$
    $\boldsymbol{I_S}, \boldsymbol{J_S} \leftarrow \text{get\_indexes}(\boldsymbol{S})$
    $I, J \leftarrow \text{index\_matching}(\boldsymbol{I_S}, \boldsymbol{I_B}, \boldsymbol{J_B})$
    $K \leftarrow \boldsymbol{J_S}$
    $\boldsymbol{C}_3 \leftarrow \boldsymbol{C}_1(I, :) + \boldsymbol{C}_1(J, :) + \boldsymbol{C}_1(K, :)$
    return $\boldsymbol{C}_3$
end function

3 Стандарт GraphBLAS

Реализация линейно-алгебраических алгоритмов подсчета и перечисления k-клик будет осуществлена с помощью стандарта GraphBLAS. Основная его концепция заключается в том, что для построения линейно-алгебраических алгоритмов на графах нужно использовать разреженные матрицы. Объясняется это тем, что размер матрицы смежности графа с n вершинами равен n^2, а матрицы инцидентности $O(n^3)$. Если граф разреженный, то при хранении вышеприведенных матриц в плотном виде мы будем аккумулировать большое количество лишней информации в качестве нулей. То есть при больших n (например, $n \sim 1$ млн.) разреженный граф в плотном виде не поместится в память компьютера, хотя в разреженном виде проблем с этим возникнуть не должно. Еще один аргумент в пользу использования разреженных матриц заключается в том, что они достаточно хорошо изучены и на настоящий момент существует большое количество высокопроизводительных параллельных алгоритмов для работы с ними.

Следующая концепция стандарта GraphBLAS заключается в том, что для линейно-алгебраических алгоритмов на графах достаточно узкого набора операций над матрицами и векторами. Преимущество такого подхода заключается в том, что небольшой набор операций хорошо оптимизируется и за счет этого достигается высокая производительность библиотек, реализующих стандарт.

Согласно [7] сделаем обзор алгоритмов и структур данных, предоставляемых стандартом GraphBLAS. Основными структурами данных в стандарте являются матрица, вектор и скаляр. Все операции, производимые над этими объектами производятся в полукольцах, то есть в кольцах, где не требуется существование обратного по сложению элемента. За счет использования операций, по структуре похожих на сложение и умножение, увеличивается спектр алгоритмов, которые можно реализовать при помощи данного стандарта.

Перечислим основные полукольца. Например, plus_times - полукольцо с стандартными операциями сложения и умножения, min_plus - полукольцо, в котором операция сложения заменена операцией взятия минимума, а операция умножения - стандартной операций сложения чисел. Также существуют полукольца max_plus, any_lt и прочие.

Стандарт поддерживает скалярное умножение матрицы на матрицу, матрицы на вектор и вектора на матрицу с скалярным произведением, которое использует операции сложения и умножения из заданного полукольца. Также поддерживаются операции поэлементного сложения и умножения матриц и векторов.

В GraphBLAS есть операция extract. Она позволяет создавать из матриц новые матрицы. Например, если задать массив-маску из индексов строк определенной матрицы, то по этой маске можно создать матрицу, в которую будут отобраны строки с индексами, представленными в маске.

Еще одной полезной операцией оказывается select. Её предназначение заключается в выборе элементов матрицы по определенному критерию. Например, с помощью этой операции можно выбрать верхний треугольник квадратной матрицы или отобрать элементы матрицы по логическому условию.

Кроме этого, удобной операцией оказывается reduce, предназначение которой это редуцирование матрицы до вектора или скаляра за счет применения определенных операций к её элементам. Например, можно редуцировать матрицу до скаляра за счет суммирования всех её элементов. Аналогично, reduce можно использовать для редуцирования вектора до скаляра.

Также стандарт GraphBLAS поддерживает такие операции как транспонирование матрицы, произведение Кронекера, извлечение индексов и значений ненулевых элементов матрицы и прочее.

4 Имплементация алгоритма

Алгоритмы подсчета и перечисления k-клик были реализованы на языке Python. Для имплементации была использована библиотека python-graphblas. Она является интерфейсом к библиотеке SuiteSparse:GraphBLAS, которая написана на языке Си. Также для реализации алгоритма 3 была использована библиотека Numba, которая позволяет компилировать Python код как код на языке Си. Вышеприведенные библиотеки поддерживают параллельные вычисления, что позволяет сделать алгоритмы подсчета и перечисления k-клик параллельными.

В алгоритме подсчета k-клик при $k \leq 6$ был использован подход без редукции матриц смежности клик, а при $k > 7$ редукция производилась. Для проведения вычислительных экспериментов использовался высокопроизводительный вычислительный кластер ”cHARISMa”(Computer of HSE for Artificial Intelligence and Supercomputer Modelling). На нем был выбран узел типа B с центральным процессором 2 x Intel Xeon Gold 6152 2.1-3.7 ГГц (2*22 ядер), и 1536 ГБ оперативной памяти. Реализация алгоритма, а также Jupyter ноутбуки с результатами вычислительных экспериментов доступны в репозитории по ссылке `https://github.com/antonov1lya/k_clique_problem`.

4.1 Проверка корректности алгоритма

Для проверки корректности алгоритма использовался граф:

$$K_5 \cup K_{10} \cup K_{15} \cup K_{20}$$

где K_n - полный граф на n вершинах. В таком графе количество k-клик вычисляется по формуле:

$$\binom{5}{k} + \binom{10}{k} + \binom{15}{k} + \binom{20}{k}$$

Приведем листинг алгоритма подсчета k-клик и значения вышеприведенной формулы при $3 \leq k \leq 20$.

```
k = 3, number of cliques = 1725, execution time = 0.00094 seconds
k = 4, number of cliques = 6425, execution time = 0.00528 seconds
k = 5, number of cliques = 18760, execution time = 0.01915 seconds
k = 6, number of cliques = 43975, execution time = 0.01931 seconds
k = 7, number of cliques = 84075, execution time = 0.04709 seconds
k = 8, number of cliques = 132450, execution time = 0.16116 seconds
k = 9, number of cliques = 172975, execution time = 0.09506 seconds
k = 10, number of cliques = 187760, execution time = 0.38944 seconds
k = 11, number of cliques = 169325, execution time = 1.48654 seconds
k = 12, number of cliques = 126425, execution time = 1.14755 seconds
k = 13, number of cliques = 77625, execution time = 4.44678 seconds
k = 14, number of cliques = 38775, execution time = 15.44357 seconds
k = 15, number of cliques = 15505, execution time = 11.35098 seconds
k = 16, number of cliques = 4845, execution time = 37.70208 seconds
k = 17, number of cliques = 1140, execution time = 30.56952 seconds
k = 18, number of cliques = 190, execution time = 21.71735 seconds
k = 19, number of cliques = 20, execution time = 63.1559 seconds
k = 20, number of cliques = 1, execution time = 78.0779 seconds
```

Рис. 1: Листинг алгоритма подсчета k-клик

```
k = 3, number of cliques = 1725
k = 4, number of cliques = 6425
k = 5, number of cliques = 18760
k = 6, number of cliques = 43975
k = 7, number of cliques = 84075
k = 8, number of cliques = 132450
k = 9, number of cliques = 172975
k = 10, number of cliques = 187760
k = 11, number of cliques = 169325
k = 12, number of cliques = 126425
k = 13, number of cliques = 77625
k = 14, number of cliques = 38775
k = 15, number of cliques = 15505
k = 16, number of cliques = 4845
k = 17, number of cliques = 1140
k = 18, number of cliques = 190
k = 19, number of cliques = 20
k = 20, number of cliques = 1
```

Рис. 2: Аналитически подсчитанное количество k-клик

Вышеприведенные значения для k-клик совпадают с аналитическими значениями, а значит можно сделать вывод о том, что алгоритм подсчета k-клик реализован корретно.

В вышеприведенном репозитории Jupyter ноутбук checking_correctness.ipynb содержит результат запуска алгоритма перечисления k-клик при $3 \leq k \leq 20$. Приведем некоторые полученные матрицы инцидентности k-клик вершинам графа.

k = 4, execution time = 0.01519 seconds

: M_{810}

	nvals	nrows	ncols	dtype	format
gb.Matrix	25700	6425	50	BOOL	csr (iso)

	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43	44	45	46	47	48	49
0	True	True	True	True							...										
1	True	True	True		True						...										
2	True	True		True	True						...										
3	True		True	True	True						...										
4		True	True	True	True						...										
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6420											...						True	True	True	True	
6421											...						True	True	True		True
6422											...						True	True		True	True
6423											...						True		True	True	True
6424											...							True	True	True	True

Рис. 3: Матрица инцидентности 4-клик вершинам графа

k = 11, execution time = 0.23754 seconds

: M_{1070}

	nvals	nrows	ncols	dtype	format
gb.Matrix	1862575	169325	50	BOOL	csr (iso)

	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43	44	45	46	47	48	49
0											...										
1											...										
2											...										
3											...										
4											...										
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
169320											...	True	True		True	True	True	True	True	True	True
169321											...	True		True	True	True	True	True	True	True	True
169322											...		True	True	True	True	True	True	True	True	True
169323											...	True	True	True	True	True	True	True	True	True	True
169324											...	True	True	True	True	True	True	True	True	True	True

Рис. 4: Матрица инцидентности 11-клик вершинам графа

Убедиться в корректности данных матриц можно, увидев, что количество строк в них совпадает с аналитически вычисленным количеством k-клик, а количество значений равно количеству k-клик, умноженному на k.

4.2 Вычислительные эксперименты

Для проведения вычислительных экспериментов были взяты наборы данных ca-CondMat, ca-HepTh, com-DBLP, com-Youtube, com-Amazon из коллекции [11]. Приведем полученные результаты для каждого набора данных.

4.2.1 Набор данных ca-CondMat

Данный граф состоит из 23.133 вершин, 93.439 ребер, а также является простым и неориентированным. Были получены следующие результаты:

k	время выполнения алгоритма (в секундах)	количество k-клик
3	0.09514	173361
4	0.09265	294008
5	0.17513	511088
6	0.10564	919604
7	12.09788	1683632
8	18.24294	3006375
9	12.54492	5006217
10	38.68056	7531707
11	60.91829	10053369
12	41.92461	11793165
13	123.24338	12095295
14	211.04703	10808741
15	146.86172	8388387
16	455.99655	5630748

При $k = 17$ алгоритму не хватило оперативной памяти для хранения матрицы смежности 6 на 6 клик.

4.2.2 Набор данных ca-HepTh

Данный граф состоит из 9.875 вершин, 25.973 ребер, а также является простым и неориентированным. Были получены следующие результаты:

k	время выполнения алгоритма (в секундах)	количество k-клик
3	0.00823	28339
4	0.04108	65592
5	0.12409	279547
6	0.07077	1123584
7	1.26139	3879079

k	время выполнения алгоритма (в секундах)	количество k-клик
8	0.49406	11532951
9	0.34562	29742628
10	1.92505	66918591
11	6.55491	131948922
12	5.32803	228841314
13	41.32701	350100366

При $k = 14$ алгоритму не хватило оперативной памяти для хранения матрицы смежности 5 на 5 клик.

4.2.3 Набор данных com-DBLP

Данный граф состоит из 317.080 вершин, 1.049.866 ребер, а также является простым и неориентированным. Были получены следующие результаты:

k	время выполнения алгоритма (в секундах)	количество k-клик
3	0.16375	2224385
4	0.42499	16713192
5	1.91298	262663639
6	41.86686	4221802226
7	1461.2348	60913718813

При $k = 8$ алгоритму не хватило оперативной памяти для хранения матрицы смежности 3 на 3 клик. Также отметим, что подсчитанные значения для k-клик совпали с результатами полученными в статье [17].

4.2.4 Набор данных com-Amazon

Данный граф состоит из 334.863 вершин, 925.872 ребер, а также является простым и неориентированным. Были получены следующие результаты:

k	время выполнения алгоритма (в секундах)	количество k-клик
3	0.04062	667129
4	0.2461	275961

k	время выполнения алгоритма (в секундах)	количество k-клик
5	0.29171	61551
6	0.21482	5799
7	345.15958	32
8	99.16535	0

4.2.5 Набор данных com-Youtube

Данный граф состоит из 1.134.890 вершин, 2.987.624 ребер, а также является простым и неориентированным. Были получены следующие результаты:

k	время выполнения алгоритма (в секундах)	количество k-клик
3	0.22746	3056386
4	7.05591	4986965
5	38.52922	7211947
6	35.98244	8443803
7	4239.05504	7959704

При $k = 8$ также как и на наборе com-DBLP алгоритму не хватило оперативной памяти для хранения матрицы смежности 3 на 3 клик.

Заключение

В данной работе на основе идеи Нешетрила-Поляка [13] и алгоритма из работы [3] были предложены алгоритмы подсчета и перечисления k-клик. Также была доказана теорема 1.7 за счет которой использование данных алгоритмов становится возможным. Эти алгоритмы были реализованы на языке Python с помощью высокопроизводительной библиотеки python-graphblas, в основе которой лежит написанная на языке Си библиотека SuiteSparse:GraphBLAS. Затем с помощью высокопроизводительного вычислительного кластера "cHARISMa" (Computer of HSE for Artificial Intelligence and Supercomputer Modelling) были произведены вычислительные эксперименты над алгоритмом. Для этого использовался узел типа B с центральным процессором 2 x Intel Xeon Gold 6152 2.1-3.7 ГГц (2*22 ядер), и 1536 ГБ оперативной памяти.

На наборах данных ca-CondMat, ca-HepTh, com-DBLP, com-Youtube, com-Amazon из коллекции [11] алгоритм хорошо себя показал в работе. Так, например, на наборе ca-CondMat посчитались клики размера $3 - 16$, на наборе ca-HepTh клики размера $3 - 13$, на наборе com-Amazon клики размера $3 - 8$ и на наборах com-DBLP и com-Youtube клики размера $3 - 7$. Однако на подсчет клик больших размеров зачастую не хватало оперативной памяти. Причем значения k для которых можно посчитать количество k-клик текущим алгоритмом зависят от количества вершин и плотности графа.

Список использованной литературы

[1] Maximilien Danisch, Oana Balalau, and Mauro Sozio. Listing k-cliques in sparse real-world graphs*. *Proceedings of the 2018 World Wide Web Conference*, 2018.

[2] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

[3] Maxim Emelin, Ilya Khlystov, Dmitriy Malyshev, and Olga Razvenskaya. On linear algebraic algorithms for the subgraph matching problem and its variants. *Optimization Letters*, 04 2023.

[4] Irene Finocchi, Marco Finocchi, and Emanuele Fusco. Clique counting in mapreduce. *Journal of Experimental Algorithmics*, 20:1–20, 10 2015.

[5] Vitaliy Gleyzer, Andrew J. Soszynski, and Edward K. Kao. Leveraging linear algebra to count and enumerate simple subgraphs. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, 2020.

[6] Shweta Jain and C. Seshadhri. The power of pivoting for exact clique counting. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

[7] Jeremy Kepner. Graphblas mathematics - provisional release 1.0 -. 2017.

[8] Jeremy Kepner and John Gilbert. *Graph Algorithms in the Language of Linear Algebra*. Society for Industrial and Applied Mathematics, USA, 2011.

[9] Longbin Lai, Wenjie Zhang, Ying Zhang, Zhengping Qian, Jingren Zhou, Zhu Qing, Zhengyi Yang, Xin Jin, Zhengmin Lai, Ran Wang, Kongzhang Hao, Xuemin Lin, and Lu Qin. Distributed subgraph matching on timely dataflow. *Proceedings of the VLDB Endowment*, 12:1099–1112, 06 2019.

[10] Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.*, 407:458–473, 2008.

[11] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[12] Amine Mhedhbi and Semih Salihoglu. Optimizing subgraph queries by combining binary and worst-case optimal joins. *Proc. VLDB Endow.*, 12:1692–1704, 2019.

[13] Jaroslav Nesetril and Svatopluk Poljak. On the complexity of the subgraph problem. 1985.

[14] Christos H. Papadimitriou and Mihalis Yannakakis. The clique problem for planar graphs. *Information Processing Letters*, 13(4):131–133, 1981.

[15] Jessica Shi, Laxman Dhulipala, and Julian Shun. *Parallel Clique Counting and Peeling Algorithms*, pages 135–146. 01 2021.

[16] Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109:254–257, 2009.

[17] Xiaowei Ye, Rong-Hua Li, Qiangqiang Dai, Hongzhi Chen, and Guoren Wang. Lightning fast and space efficient k-clique counting. pages 1191–1202, 2022.