Санкт-Петербургский государственный университет

Кафедра системного программирования

Подсчет треугольников. Алгоритмы Sandia и Burkhardt

Мелашенко Ксения, Нурмухаметов Рафик, Сыресенков Илья
Группа 22.Б15-мм

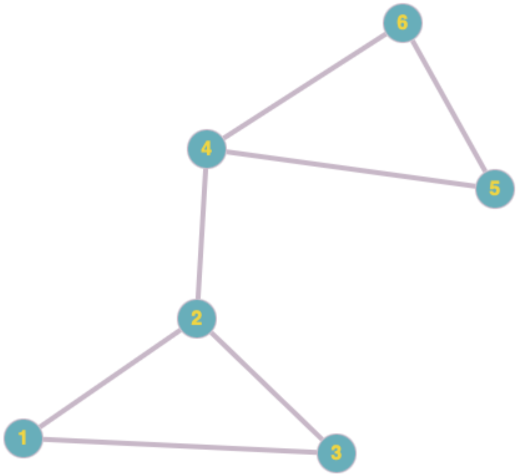
Санкт-Петербург
2025

Постановка задачи

Задача: дан неориентированный граф $G\,(V, E)$ без петель и кратных ребер. Требуется найти **количество треугольников в графе**

- Треугольником будем называть тройку уникальных вершин u, v, w, попарно соединенных ребрами (u, v), (u, w), $(v, w) \in E$, где E — множество ребер

Пример



Определения

- $\times$ — умножение матриц
- $\otimes$ — поэлементное умножение матриц
- $\sum$ — сумма всех элементов
- A — булева симметричная матрица смежности входного графа
- A^3 — все пути длины 3.
- $trace$ — сумма всех элементов главной диагонали, $trace(A) = \sum a_{ii}$
- L — верхнетреугольная матрица от A,
- U — нижнетреугольная матрица от A
- $A = L + U$

Базовая версия матричного алгоритма

Количество треугольников в графе выражается формулой, использующей сумму всех элементов главной диагонали:

$$\frac{trace(A^3)}{6}$$

Значение $A^3[i][i]$ отображает количество путей, начинающийся и заканчивающийся в вершине i, что и есть количество треугольников, проходящих через вершину i.
Так как число треугольников считается для каждой вершины и каждый треугольник будет посчитан трижды, то общее количество необходимо разделить на 3. Также, так как граф неориентированный, делим еще на 2.

Алгоритм Burkhardt

В базовой версии матричного алгоритма последнее умножение A^3 можно заменить вычислительно менее "тяжелым" поэлементным умножением $A^2 \otimes A$. Получаем

$$\frac{\sum(A^2 \otimes A)}{6}$$

Алгоритм Sandia

Данный алгоритм является последним по времени разработки и является наиболее производительным

$$\sum((U \times U) \otimes U)$$

Каждый треугольник будет посчитан единожды.

Набор данных

Графы взяты из Stanford Network Analysis Platform (SNAP)[1], где представлены различные датасеты графов, описывающие связи людей в социальных сетях, цитирования научных работ, перекрестки на дорогах и т.д

Возьмем следующие графы:

- Неориентированные
- Невзвешенные
- Без кратных рёбер

[1] https://snap.stanford.edu/data/

Выбранные графы

Имя	Вершины	Рёбра	Описание
roadNet-PA	1 088 092	1 541 898	Дорожная сеть штата Пенсильвания
roadNet-CA	1 965 206	2 766 607	Дорожная сеть штата Калифорния
com-Amazon	334 863	925 872	Сеть продуктов Amazon
com-DBLP	317 080	1 049 866	Сеть со-авторства статей DBLP
email-Enron	36 692	183 831	Граф взаимодействия электронных почт от Enron
ca-CondMat	23 133	93 497	Граф со-авторства статей из Arxiv Condensed Matter
ca-AstroPh	18 772	198 110	Граф со-авторства статей из Arxiv Astro Physics
ego-Facebook	4 039	88 234	Граф соцсети Facebook

Эксперименты

Сравнение средней работы реализаций алгоритмов Sandia и Burkhardt на библиотках SuiteSparse:GraphBLAS и SPLA

- **Цель 1:** Узнать, на какой их двух библиотек быстрее реализация алгоритма Sandia.
- **Цель 2:** Узнать, на какой их двух библиотек быстрее реализация алгоритма Burkhardt.

Эксперименты

Ход эксперимента

- Провести по 20 запусков алгоритмов на каждом графе
- Рассчитать среднее время выполнения алгоритмов на каждом графе в мс (1 с = 1 000 мс)
- Построить доверительные интервалы

Все запуски будут проводиться на CPU

SuiteSparse:GraphBLAS и SPLA

- A.mxm(A) — умножение матриц
- A.ewise_mult(A) — поэлементное умножение матриц
- A.reduce(”sum”) — суммирование всех значений в матрице

Эксперименты

Характеристики вычислительной машины:

- ОС MacOS Sequoia 15.6.1
- Процессор Apple M2
  - 8 ядер CPU
  - L1-кэш 256 KB
  - L2-кэш 20MB
  - L3-кэш 8 MB
- 8GB RAM
- 256GB SSD

Эксперименты

Почему не Apache Spark?
Сложно/не очевидно, как сделать «непосредственно» реализации алгоритмов, описанных в этой презентации.

Более контекстный подход в Apache Spark

- Для каждой вершины v собрать всех соседей
- Для каждого v построить все пары (u, w) из его соседей
- Проверить, существует ли ребро (u, w)

Есть MLlib, но он, для других задач.