Экспериментальный анализ

Автор: Погожельская Влада

23 сентября

Исходные данные:
Графы LUBM300, LUBM500, LUBM1M, LUBM1.5M, LUBM1.9M, были взяты без изменений; регулярные выражения взяты из архива RefinedDataForRPQ, были преобразованы для удобного распознавания их библиотекой pyformlang. Так как к данной группе графов предлагался одинаковый набор запросов, были взяты регулярные выражения вида 'q*0'.

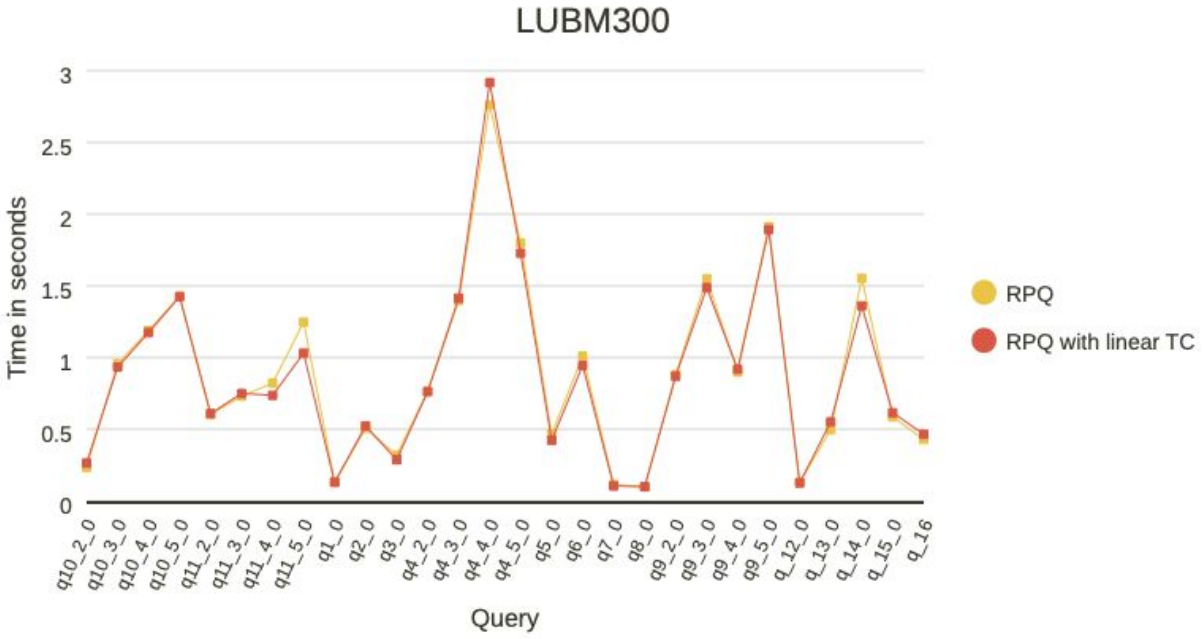
Оборудование:
Замеры производились на ноутбуке с процессором 1,4ГГц, 4-ядерный Intel Core i5 8-го поколения, 128МБ eDRAM, 8ГБ LPDDR3 2133 МГц RAM

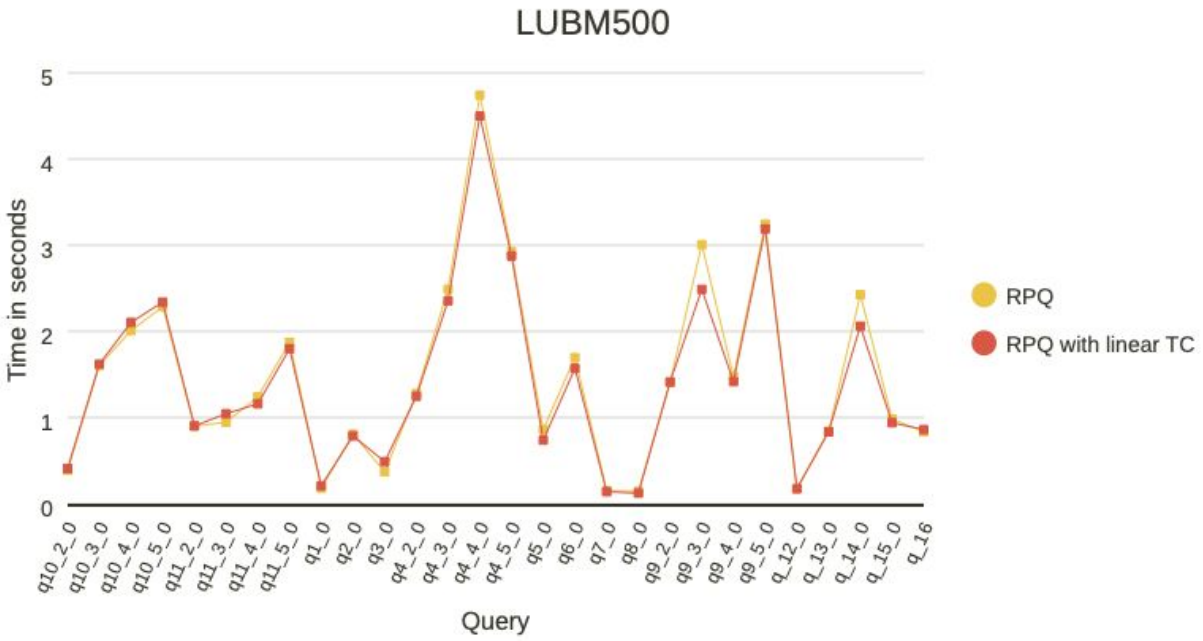
ОС: macOS Catalina 10.15.6

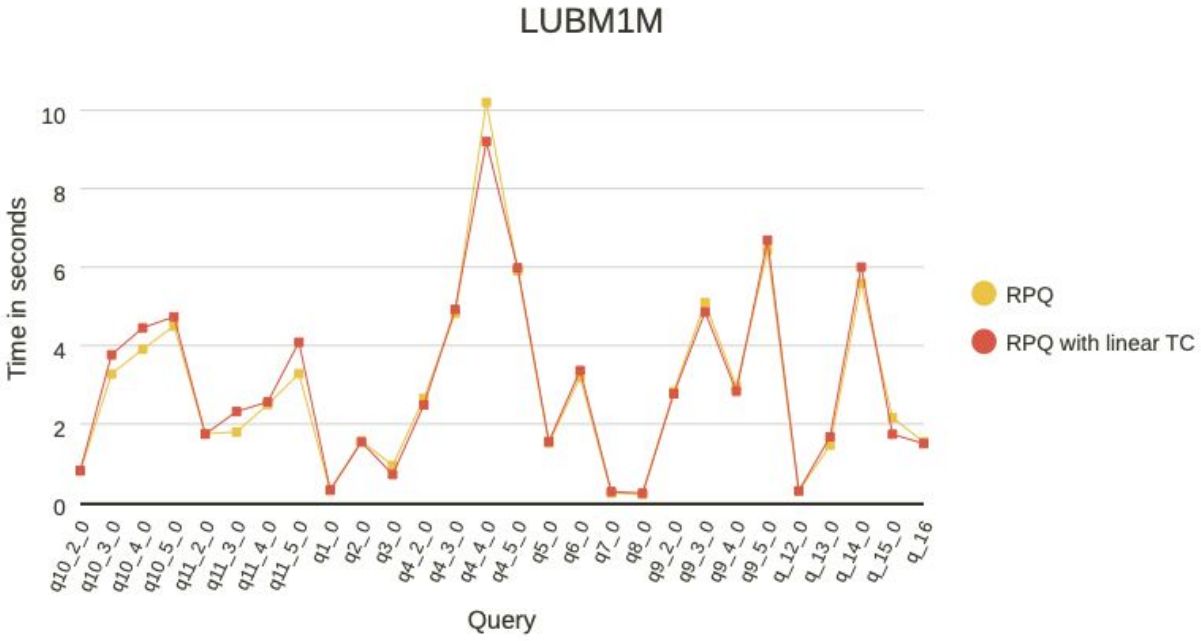
RAM, выделенная для Docker: 6Gb

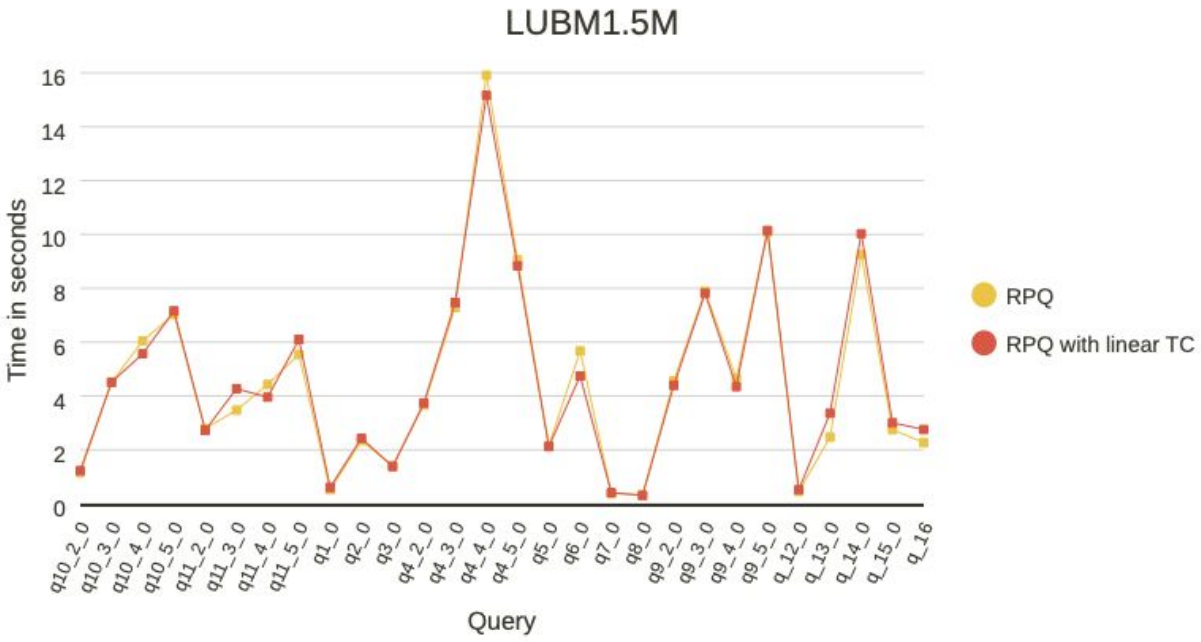
Ход эксперимента:
При проведении эксперимента по производительности пересечения автоматов с применением двух методов поиска транзитивного замыкания(с помощью умножения на матрицу смежности и с помощью возведения матрицы в квадрат) для каждого замера алгоритмы запускались по 5 раз, в качестве результатов бралось среднее значение. Для вычисления времени работы алгоритмов использовалась функция default_timer() из библиотеки timeit.

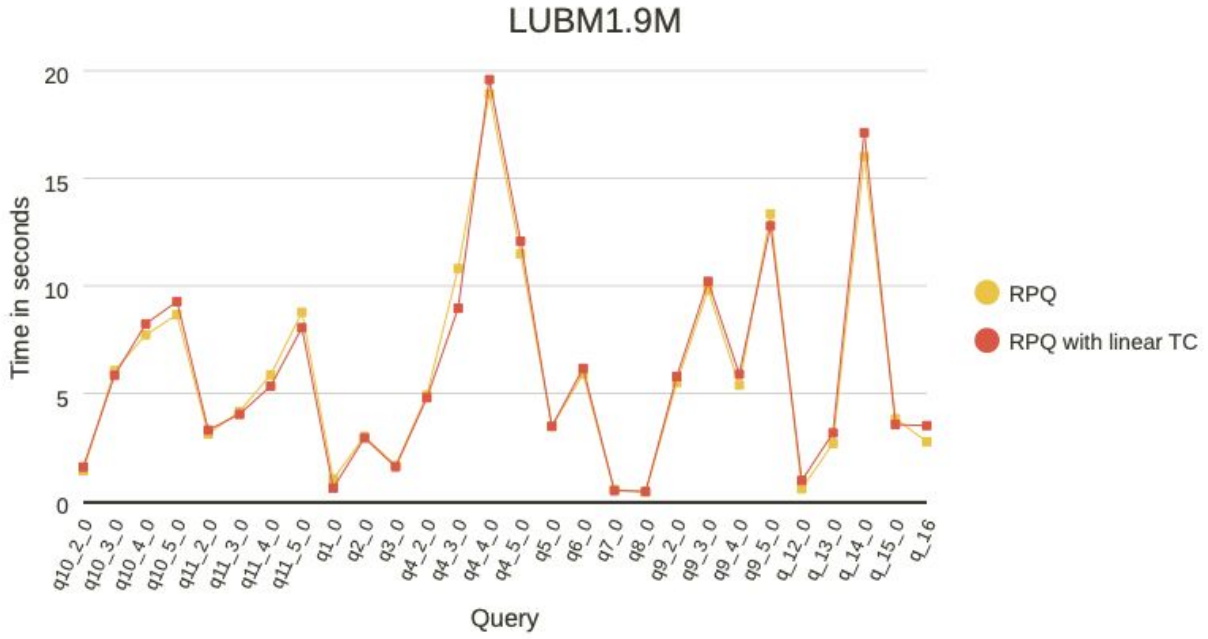
На графиках представлены результаты запусков реализации, основанной на возведении в квадрат(RPQ) и основанной на умножении на матрицу смежности(RPQ with linear TC) в удобном для сравнения формате.


LUBM300
Time in seconds
3
2.5
2
1.5
1
0.5
0
q10_2_0
q10_3_0
q10_4_0
q10_5_0
q11_2_0
q11_3_0
q11_4_0
q11_5_0
q1_0
q2_0
q3_0
q4_2_0
q4_3_0
q4_4_0
q4_5_0
q5_0
q6_0
q7_0
q8_0
q9_2_0
q9_3_0
q9_4_0
q9_5_0
q_12_0
q_13_0
q_14_0
q_15_0
q_16
Query
RPQ
RPQ with linear TC


LUBM500
Time in seconds
5
4
3
2
1
0
q10_2_0
q10_3_0
q10_4_0
q10_5_0
q11_2_0
q11_3_0
q11_4_0
q11_5_0
q1_0
q2_0
q3_0
q4_2_0
q4_3_0
q4_4_0
q4_5_0
q5_0
q6_0
q7_0
q8_0
q9_2_0
q9_3_0
q9_4_0
q9_5_0
q_12_0
q_13_0
q_14_0
q_15_0
q_16
Query
RPQ
RPQ with linear TC


LUBM1M
Time in seconds
0
2
4
6
8
10
q10_2_0
q10_3_0
q10_4_0
q10_5_0
q11_2_0
q11_3_0
q11_4_0
q11_5_0
q1_0
q2_0
q3_0
q4_2_0
q4_3_0
q4_4_0
q4_5_0
q5_0
q6_0
q7_0
q8_0
q9_2_0
q9_3_0
q9_4_0
q9_5_0
q_12_0
q_13_0
q_14_0
q_15_0
q_16
Query
RPQ
RPQ with linear TC


LUBM1.5M
Time in seconds
0
2
4
6
8
10
12
14
16
q10_2_0
q10_3_0
q10_4_0
q10_5_0
q11_2_0
q11_3_0
q11_4_0
q11_5_0
q1_0
q2_0
q3_0
q4_2_0
q4_3_0
q4_4_0
q4_5_0
q5_0
q6_0
q7_0
q8_0
q9_2_0
q9_3_0
q9_4_0
q9_5_0
q_12_0
q_13_0
q_14_0
q_15_0
q_16
Query
RPQ
RPQ with linear TC



Время выведения достижимых в полученном графе пар во всех замерах составило меньше полусекунды.

Контрольные значения (количество достижимых в полученном графе пар) во всех случаях совпали. Ниже приведена таблица с результатами некоторых из них.

Граф	Запрос	Количество пар
LUMB300	q10_2_0	7797
LUMB300	q10_5_0	402517
LUMB300	q3_0	6027
LUMB500	q10_2_0	12993
LUMB500	q10_5_0	670881
LUMB500	q3_0	9989
LUMB1M	q10_2_0	26098
LUMB1M	q10_5_0	1338293

LUMB1M	q3_0	20252
LUMB1.5M	q10_2_0	39071
LUMB1.5M	q10_5_0	2006057
LUMB1.5M	q3_0	30334
LUMB1.9M	q10_2_0	50606
LUMB1.9M	q10_5_0	2600201
LUMB1.9M	q3_0	39246

Выводы:

Результаты замеров производительности двух реализаций не показали существенных различий. Основным предположением для объяснения такого результата является то, что, так как поддерживаемые библиотекой pygraphblas операции оптимизированы для работы с большими разреженными матрицами, то это позволяет сравняться алгоритмам по времени: маленькое количество возведений более плотной матрицы в квадрат компенсируется большим количество умножений на разреженную матрицу смежности. Из полученных результатов времени работы алгоритмов можно сделать выводы о том, что pygraphblas является подходящей библиотекой для работы с разреженными матрицами по крайней мере в размерах до 2M вершин и, думаю, что при реализации данных алгоритмов с помощью другой библиотеки, не оптимизированной для работы с ними, разница в производительности была бы более ощутимой. Также остается неизвестным поведение алгоритмов при запуске на более крупных графах в силу недостаточной вычислительной мощности ноутбука.