**УДК** Номер <http://ofernio.ru/portal/grnti.php?itemMenu=classification>

**Вычисление контекстно-свободных запросов к графам с семантикой всех путей с помощью матричных операций**

**Рустам Шухратуллович Азимов, Семён Вячеславович Григорьев**

**Аннотация**

**Предмет исследования.** Рассмотрена задача вычисления контекстно-свободных запросов к помеченным графам. Данная задача заключается в поиске различных путей графа, метки на рёбрах которых образуют слова из языка, порождённого входной контекстно-свободной грамматикой. Существует два наиболее эффективных подхода к решению данной задачи с использованием операций линейной алгебры: с использованием обычного матричного умножения и с использованием произведения Кронекера. Но до сих пор нельзя в полной мере сравнить эти два подхода, так как не существует алгоритма, использующего обычное матричное произведение, способного обрабатывать контекстно-свободные запросы с самой сложной семантикой запроса всех путей, в которой требуется предоставить все пути, соответствующие запросу. В работе предложена модификация алгоритма вычисления контекстно-свободных запросов к графам, использующего обычное матричное произведение, которая способна обрабатывать запросы с семантикой всех путей. **Метод.** В матрице смежности входного графа для каждой пары вершин храним дополнительную информацию о найденных путях между этими вершинами в виде множества возможных промежуточных вершин. На первом этапе осуществляется построение множества матриц, хранящих в себе такую информацию о всех путях, удовлетворяющих входному запросу. На втором этапе осуществляется восстановление всех запрашиваемых путей. **Основные результаты.** Предложенный алгоритм был реализован и было проведено сравнение с другими наиболее эффективными алгоритмами вычисления контекстно-свободных запросов. Результаты экспериментального исследования показали, что предложенный алгоритм существенно эффективнее восстанавливает запрашиваемые пути, однако в некоторых случаях потребляет существенно больший объём памяти, чем алгоритм, основанный на произведении Кронекера. **Практическая значимость.** Предложенный алгоритм может быть применён в задачах статического анализа кода, биоинформатике, сетевом анализе, а также в графовых базах данных, когда требуется найти все возможные зависимости в данных, представленных в виде помеченного графа.

**Ключевые слова**

контекстно-свободные запросы к графам, линейная алгебра, контекстно-свободные грамматики, матричное умножение, графовые базы данных, стандарт GraphBLAS

**Благодарности**

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-37-90101.

**Computing the context-free queries to graphs with all-path query semantics using matrix operations**

**Rustam Azimov, Semyon Grigorev**

**Abstract**

**Subject of Research.** The problem of context-free path querying is considered. This problem consists in finding various paths of the graph, the labels on the edges of which form words from the language generated by the input context-free grammar. There are two approaches to evaluate context-free path queries using linear algebra operations: matrix multiplication-based and the Kronecker product-based. But until now, it is impossible to fully compare these two approaches, since there is no algorithm using the matrix multiplication capable of handling context-free path queries with the most complex all-path query semantics, in which the all paths that match the query must be provided. The paper proposes a modification of the algorithm for computing context-free path queries using the matrix multiplication, which is capable of processing queries with the all-path query semantic. **Method.** In the adjacency matrix of the input graph for each pair of vertices, we store additional information about the paths found between these vertices in the form of a set of possible intermediate vertices. At the first stage, a set of matrices is constructed that store such information about all paths that satisfy the input query. At the second stage, all queried paths are restored from the constructed set of matrices. **Main Results.** The proposed algorithm was implemented and a comparison was made with other most efficient algorithms for evaluating context-free path queries. The results of the experimental study showed that the proposed algorithm is significantly more efficient in restoring the queried paths, but in some cases it consumes a significantly larger amount of memory than the algorithm based on the Kronecker product. **Practical Relevance.** The proposed algorithm can be applied in problems of static code analysis, bioinformatics, network analysis, as well as in graph databases, when it is required to find all possible dependencies in the data presented in the form of a labeled graph.

**Keywords**

context-free path querying, linear algebra, context-free grammars, matrix multiplication, graph databases, GraphBLAS

**Aknowledgements**

The reported study was funded by RFBR, project number 19-37-90101.

**Введение**

В современном мире становится всё больше данных, которые требуют обработки и анализа. При этом графы являются одной из самых распространённых и удобных структур данных, позволяя компактно представлять большие объемы информации и реализовывать алгоритмы для извлечения из данных различной интересной информации — существование определенных путей, достижимость некоторых вершин и т.д. Графы используются в статическом анализе кода [1,2], биоинформатике [3], в сетевом анализе [4]. Также, в настоящее время активно развиваются графовые базы данных (Neo4j, RedisGraph и т.д.), используемые для хранения и реализации запросов к данным в виде графов.

Одной из важнейших графовых задач является поиск различных путей в графах. Чтобы описать свойства искомых путей, обычно задают специальные ограничения. Эти ограничения формулируются в виде запроса к графу, а ответом на такой запрос является информация о существовании путей, удовлетворяющих данным ограничениям. Семантику таких запросов называют реляционной. Кроме того, часто, требуется предъявить все или хотя бы один из таких путей, и тогда говорят, что такие запросы вычислены в семантике всех или одного пути соответственно.

Для описания ограничений на пути в помеченном графе естественно использовать формальные грамматики над некоторым алфавитом. С помощью такой грамматики ограничивают множество слов, получаемых конкатенацией меток на рёбрах рассматриваемых путей. В настоящее время активно исследуются ограничения, представленные в виде контекстно-свободных (КС) грамматик. Этот подход позволяет описывать более широкий класс запросов (КС-запросов), чем, например, широко используемые на практике регулярные выражения. Однако большинство существующих алгоритмов вычисления КС-запросов имеют на больших графах низкую производительность, что затрудняет их применение на практике.

Был предложен ряд алгоритмов вычисления КС-запросов к графам, основанных на различных методах синтаксического анализа: на основе (G)LL и (G)LR алгоритмов[5] [6] [7] [8]; на основе CYK алгоритма [4]; с использованием парсер-комбинаторов [9]. Но недавнее исследование Йохем Куиджперса (Jochem Kuijpers) [10] показывает, что существующие решения не применимы для анализа реальных графов из-за значительных времени работы и потребления памяти.

Одним из распространённых способов улучшения производительности алгоритмов на графах является их переформулирование в терминах линейной алгебры. В такой формулировке основными являются вычисления операций над матрицами и векторами. Для тех алгоритмов, которые позволяют найти такое представление, становится возможным применить для представления графов разреженные матрицы (т.е. матрицы, имеющие малое количество рёбер) и использовать параллельные вычисления, в частности, на основе GPU-технологий. Кроме того, такого рода алгоритмы зачастую просты в реализации, так как позволяют использовать существующие библиотеки линейной алгебры (GraphBLAS::SuiteSparse, cuSPARSE, cuBLAS, m4ri, Scipy и др.). Относительно недавно такая формулировка была найдена и для задач вычисления КС-запросов к графам, а именно, для реляционной семантики запросов [11] и для семантики одного пути [12]. Данные алгоритмы основаны на матричных операциях и позволяют получить достаточно высокоэффективные реализации с помощью существующих библиотек линейной алгебры. Другой алгоритм вычисления КС-запросов к графам, сформулированный на языке линейной алгебры, основан на произведении Кронекера [13]. Этот алгоритм принципиально отличается от алгоритмов, основанных на обычных матричных операциях, тем, что обрабатывает КС-запросы в виде произвольных КС-грамматик без необходимости преобразовывать их в нормальную форму, в то время как прочие алгоритмы требуют преобразовывать входную КС-грамматику в бинарную нормальную форму, что приводит к увеличению размеров КС-грамматик и негативно влияет на скорость выполнения КС-запросов. Кроме того, алгоритм основанный на произведении Кронекера строит в процессе вычисления КС-запросов строит более сложные структуры, тем самым позволяет обрабатывать КС-запросы любой из трёх семантик. Но до сих пор нельзя в полной мере сравнить эти два подхода, так как не существует алгоритма, использующего обычное матричное произведение, способного обрабатывать КС-запросы с самой сложной семантикой всех путей.

Таким образом, в данной работе ставится задача разработки алгоритма вычисления КС-запросов, основанного на матричном произведении и способного обрабатывать запросы с семантикой всех путей.

**Постановка задачи**

Пусть Σ — конечное множество терминальных символов. *Помеченным графом* будем называть тройку *D* = (*V, E,* Σ), где *V* является множеством вершин, а — множеством ребер с метками из алфавита Σ. Для пути *π* в графе *D* мы будем использовать обозначение *l*(π), чтобы указать на слово, полученное конкатенацией меток на ребрах данного пути. Кроме того, запись *mπn* будет обозначать, что в рассматриваемом графе существует путь из вершины в вершину .

*Контекстно-свободная грамматика в слабой нормальной форме Хомского* — это четвёрка *G* = (*N,* Σ*, P, S*), где *N* — конечное множество нетерминальных символов, Σ — конечное множество терминальных символов, — стартовый нетерминал и *P* — конечное множество правил следующего вида:

* , для ;
* , для *,* , где — пустая строка.

Мы рассматриваем только КС-грамматики в такой бинарной нормальной форме, так как для каждой КС-грамматики можно построить эквивалентную ей грамматику в данной форме [14].

Мы будем писать, чтобы указать, что строка может быть получена из нетерминала *A* некоторой последовательностью применений правил КС- грамматики. А я*зыком*, задаваемым КС-грамматикой *G* = (*N,* Σ*, P, S)* будем называть

.

Для заданного графа *D* = (*V, E*) и КС-грамматики *G* = (*N,* Σ*, P, S*) определяются *контекстно-свободные отношения* , являющиеся множеством пар вершин, между которыми существует путь, образующий строку, выводимую из стартового нетерминала *S*. Такие отношения определяются следующим образом:

.

Используя данные определения мы можем сформулировать постановку задачи вычисления КС-запросов к графам с семантикой всех путей.

**Задача.** Проблема вычисления КС-запросов к графам с семантикой всех путей для заданного помеченного графа D и КС-грамматики G заключается в вычислении контекстно-свободного отношения и нахождении для каждой пары вершин всех путей между ними, таких что .

**Матричный алгоритм вычисления КС-запросов с семантикой всех путей**

Наш алгоритм основан на матричном алгоритме вычисления КС-запросов к графам [11] для реляционной семантики запросов. Этот алгоритм сводит вычисление КС-запросов к вычислению операций над булевыми матрицами и, как следствие, позволяет использовать высокопроизводительные библиотеки линейной алгебры и использовать современное параллельное оборудование для решения данной задачи анализа графов.

Мы будем хранить дополнительную информацию в используемых матрицах, чтобы иметь возможность восстанавливать все пути, которые образуют слова, выводимые из любого нетерминала заданной КС-грамматики. Для этого мы введем структуру данных , которая будет использоваться для хранения элементов матриц, описывающих найденные пути как конкатенации двух меньших путей. Этой информации достаточно для восстановления всех найденных путей. Здесь left и right — индексы стартовой и конечной вершин найденных путей, а middles — множество индексов промежуточных вершин, используемых при конкатенации двух меньших путей. Когда мы не находим путь для некоторых пар вершин (m, n), мы обозначаем это специальным элементом ⊥ = (0, 0, ∅).

Кроме того, мы будем использовать обозначение *правильной* матрицы, для каждого элемента с индексами 𝑖, 𝑗 которой этот элемент либо равен (𝑖, 𝑗, \_), либо ⊥.

Мы определим операцию ⊗ для AllPathIndexes , , которые не равны ⊥ и , как .

Если же хотя бы один из операндов равен ⊥, то .

Кроме того, мы определим бинарную операцию ⊕ для AllPathIndexes , , которые не равны ⊥, и , как

.

Если ровно один из операндов равен ⊥, то равен другому операнду. Если же оба операнда равны ⊥, то .

Используя ⊗ как умножение элементов матриц и ⊕ как сложение, мы можем определить матричное умножение, 𝑎 ⊙ 𝑏 = 𝑐, где 𝑎 и 𝑏 — матрицы подходящего размера, которые имеют AllPathIndexes в качестве элементов, как .

Также мы используем операцию + поэлементного сложения матриц 𝑎 и 𝑏 одинакового размера: 𝑎 + 𝑏 = 𝑐, где .

Используя полученные матричные операции, мы представляем матричный алгоритм вычисления КС-запросов к графам для семантики всех путей. Псевдокод данного алгоритма приведен на рис. 1.

Для заданных КС-грамматики 𝐺 = (𝑁, Σ, 𝑃, 𝑆) и помеченного графа 𝐷 = (𝑉, 𝐸, Σ) результатом работы представленного алгоритма является набор матриц 𝑇 (так называемый индекс), который хранит информацию обо всех путях в графе 𝐷, которые образуют слово, выводимое из некоторого нетерминала КС-грамматики 𝐺.

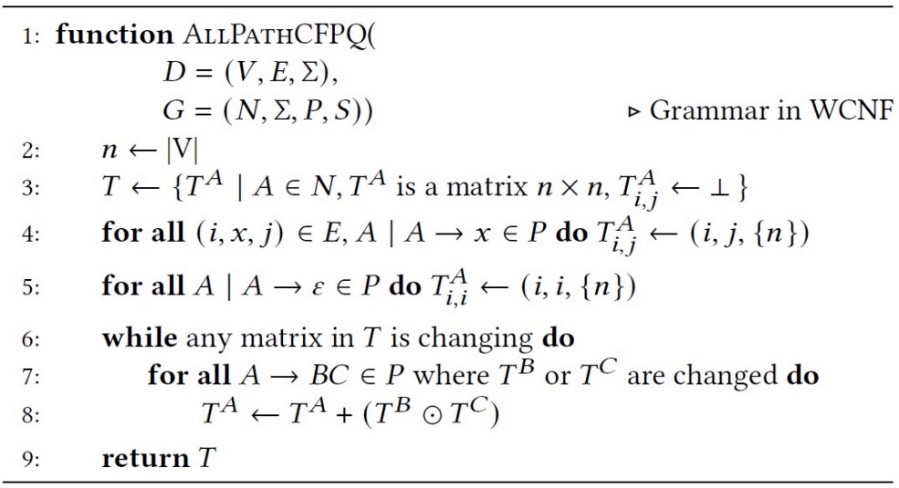


Рисунок 1. Матричный алгоритм вычисления КС-запросов к графам для семантики всех путей

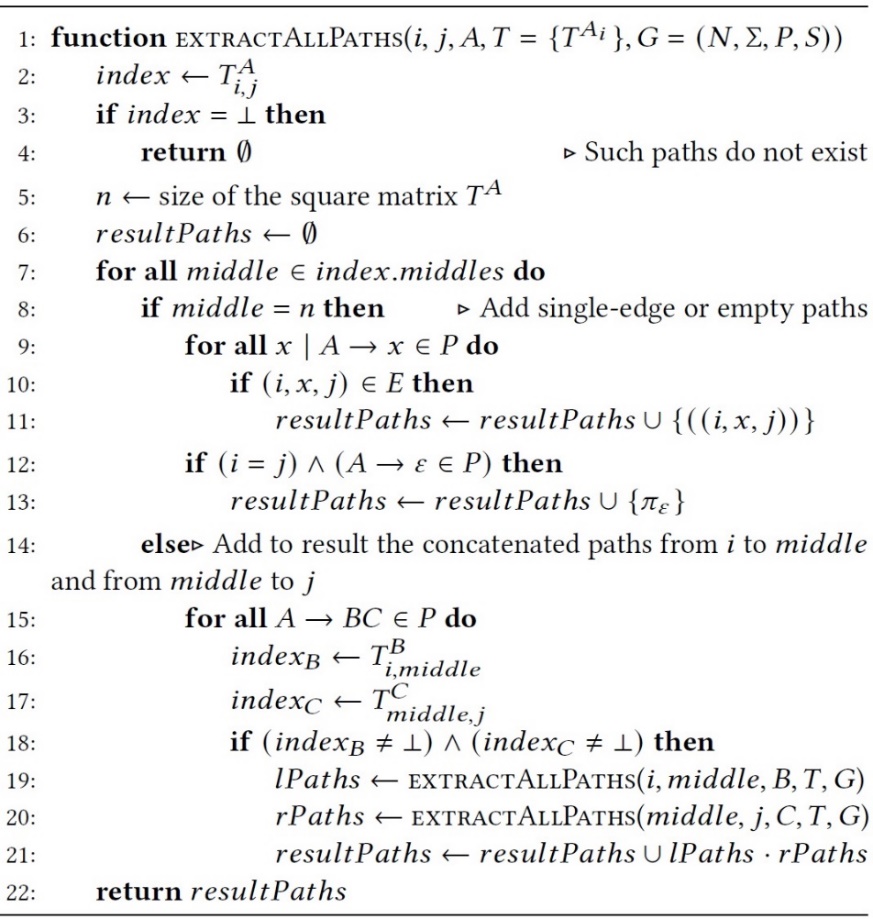
****

Рисунок 2. Алгоритм выявления всех путей

Обратите внимание, что в строке 4 мы добавляем специальное значение 𝑛 к к , чтобы указать, что это путь с одним ребром или пустой путь .

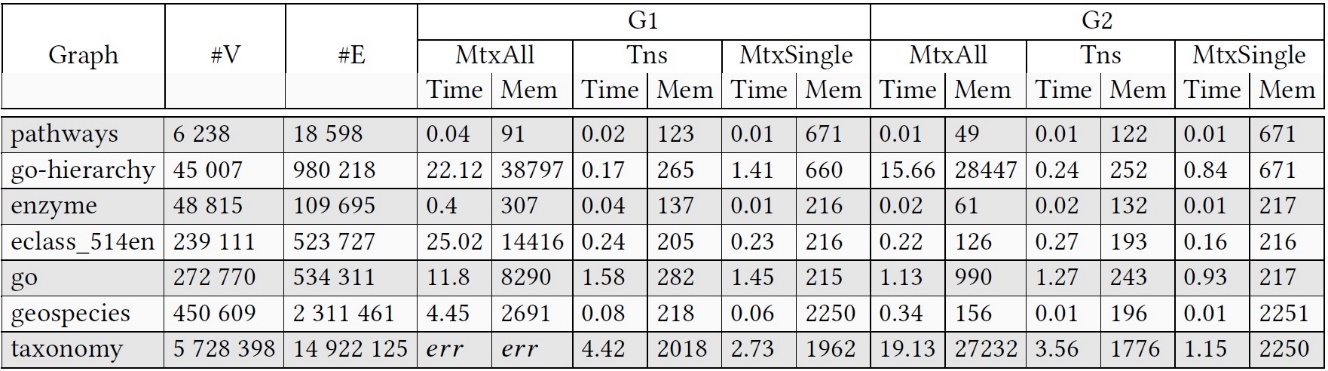
Построенный индекс позволяет вычислить контекстно-свободное отношение (включив в него все пары вершин (i,j), для которых , а также выявить все найденные пути, тем самым решить задачу вычисления КС-запросов к графам для семантики всех путей. Однако, множество таких путей может быть бесконечным в случае наличия циклов во входном графе. С практической точки зрения, при построении найденных путей необходимо использовать «ленивое» вычисление или каким-то образом ограничить результирующее множество путей. Например, можно потребовать построить некоторое фиксированное количество путей или ограничить сверху длину путей.

Кроме того, мы предлагаем алгоритм извлечения найденных путей, псевдокод которого приведен на рис. 2. Данные алгоритм возвращает множество с пустым путём , только если 𝑖 = 𝑗 и 𝐴 → 𝜀 ∈ 𝑃. Если для заданных 𝑖, 𝑗, 𝐴, соответсвующий элемент матрицы равен ⊥, то наш алгоритм возвращает пустое множество, так как путей искомого вида не существует. Обратите внимание, что в строке 19 мы используем операцию , которая естественным образом обобщает операцию конкатенации путей, строя все возможные конкатенации пар путей из двух множеств. Предполагается, что множества путей вычисляются «лениво», чтобы обеспечить завершаемость алгоритма в случае бесконечного количества путей.

**Эксперименты**

Целью данного экспериментального исследования — является изучение применимости предложенного матричного алгоритма вычисления КС-запросов к графам для семантики всех путей и сравнение с другими наиболее эффективными алгоритмами вычисления КС-запросов, использующими операции линейной алгебры. Мы сравним следующие реализации:

Таблица 1. Время построение индекса в секундах и использованная память в мегабайтах



* 𝑀𝑡𝑥𝑆𝑖𝑛𝑔𝑙𝑒 - реализация из [12] матричного алгоритма вычисления КС-запросов для семантики одного пути;
* 𝑇𝑛𝑠 - реализация из [13] алгоритма вычисления КС-запросов к графам, основанного на произведении Кронекера, для всех трёх семантик запросов;
* 𝑀𝑡𝑥𝐴𝑙𝑙 - реализация предложенного матричного алгоритма вычисления КС-запросов к графам для семантики всех путей, которая использует реализацию стандарта GraphBLAS для матричных манипуляций.

Все реализации используют ЦПУ и матрицы в разреженном формате. Сначала мы измерили время выполнения и требуемую память для создания индекса. Затем мы сравнили практическую применимость извлечения путей для обеих реализаций 𝑀𝑡𝑥𝐴𝑙𝑙 и 𝑇𝑛s, которые способны обрабатывать запросы с семантикой всех путей. Исходный код доступен на GitHub[[1]](#footnote-1). Для экспериментов мы использовали ПК с установленной ОС Ubuntu 18.04, процессором Intel Core i7-6700, 3,4 ГГц и оперативной памятью DDR4 64 ГБ.

**Данные.** Мы используем графы и соответствующие запросы из датасета, представленного в [12], который содержит реальные данные в формате RDF и запросы 𝑔1, 𝑔2, которые являются вариациями same-generation запроса [15] — важного примера реальных запросов, которые могут быть выражены с помощью КС-грамматик, но не с помощью регулярных.

**Результаты.** Результаты создания индекса для всех трех реализаций представлены в таблице 1. Мы видим, что матричный алгоритм для семантики одного пути позволяет строить индекс наиболее быстро, особенно при работе с большими графами. Но 𝑀𝑡𝑥𝑆𝑖𝑛𝑔𝑙𝑒 не может восстановить все найденные при анализе графа пути, так как использует более простой индекс, чтобы восстановить только один путь для каждой пары вершин. Однако реализация 𝑇𝑛𝑠 алгоритма, основанного на произведении Кронекера, использует более сложный, но компактный индекс, что позволяет не только восстановить все найденные пути, но и потреблять меньше памяти, чем матричные алгоритмы.

Реализация 𝑀𝑡𝑥𝐴𝑙𝑙 предлагаемого матричного алгоритма вычисления КС-запросов к графам для семантики всех путей сравнима по времени выполнения с реализацией 𝑇𝑛𝑠 на маленьких графах, но значительно медленнее на некоторых больших графах со сложной структурой. Кроме того, на некоторых графах 𝑀𝑡𝑥𝐴𝑙𝑙 потребляет значительно больше памяти, чем 𝑇𝑛𝑠. Причина такого поведения заключается в том, что предлагаемый матричный алгоритм пытается сохранить информация обо всех найденных путях в более явном виде. На самом большом графе 𝑡𝑎𝑥𝑜𝑛𝑜𝑚𝑦 и запросе 𝑔1 произошла ошибка нехватки памяти для реализации 𝑀𝑡𝑥𝐴𝑙𝑙, что отмечено в таблице с помощью «err»

После построения индекса мы сравнили время выполнения извлечения путей для.

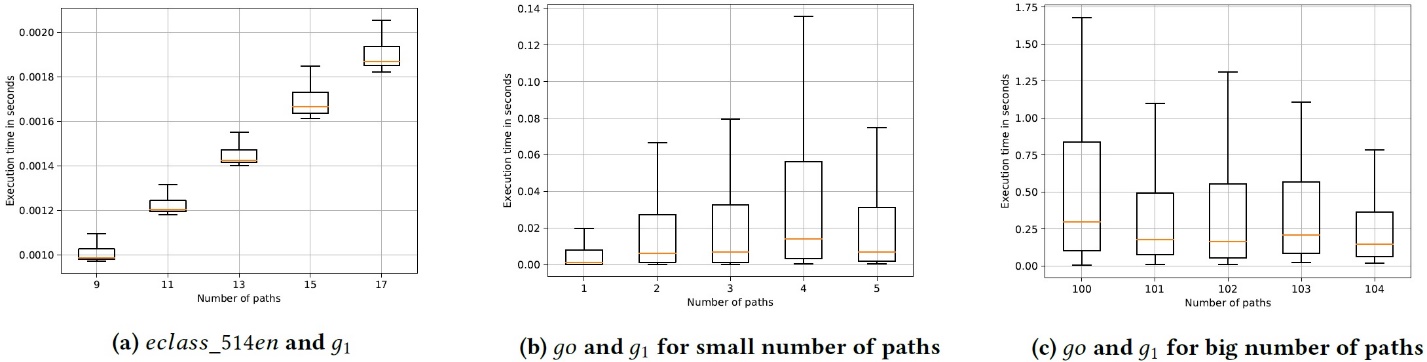
****

Рисунок 3. Время выявления путей в секундах для реализации Tns алгоритма, основанного на произведение Кронекера

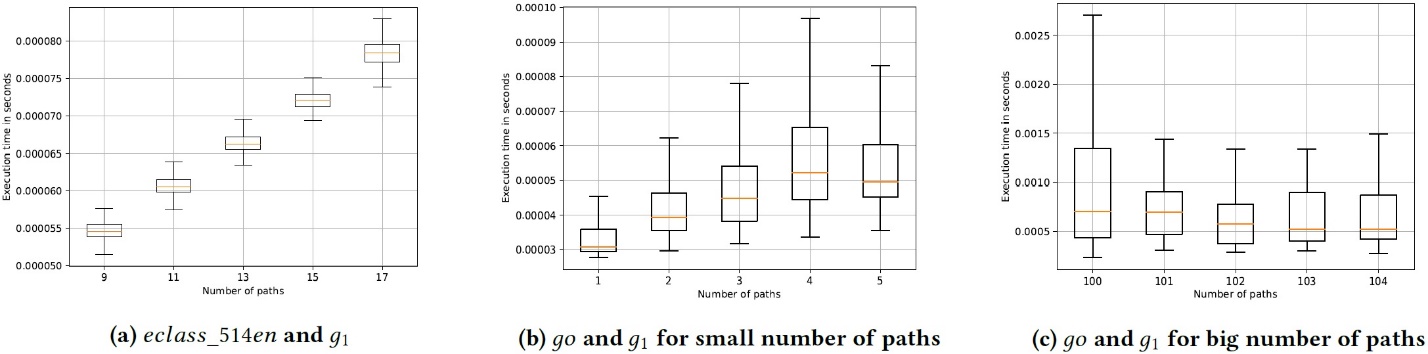
****

Рисунок 4. Время выявления путей в секундах для реализации MtxAll предложенного матричного алгоритма

двух реализаций, поддерживающих семантику запроса всех путей. Результаты извлечения путей для графов 𝑔𝑜 и 𝑒𝑐𝑙𝑎𝑠𝑠\_514𝑒𝑛 представлены на рис. 3 и 4 (ящики с усами стандартные, указаны медианы и выбросы опускаются). Для завершения вычислений мы ограничиваем максимальную длина путей до 10. После этого мы извлекаем пути для каждого пары вершин и группируем время извлечения по количеству возвращаемых путей. Мы видим, что время извлечения путей в реализации 𝑀𝑡𝑥𝐴𝑙𝑙 предложенного матричного алгоритма до 1000 раз быстрее, чем для реализации 𝑇𝑛𝑠. Как было сказано выше, в предложенном матричном алгоритме, мы строим индекс с более явной информацией о найденных путях, что позволяет значительно быстрее их восстанавливать.

**Выводы.** По результатам экспериментального исследования мы можем сделать следующие выводы.

* Для семантики одного пути, когда требуется предъявить только один путь для каждой пары вершин, матричный алгоритм вычисления КС-запросов к графам [12] является наиболее производительным, а алгоритм [13], основанный на произведении Кронекера, потребляет наименьшее количество памяти.
* Для семантики всех путей предложенный матричный алгоритм и алгоритм, основанный на произведении Кронекера, сопоставляются следующим образом. Если необходимо часто пересчитывать индекс для изменяющегося графа или запроса, то лучший выбор — алгоритм [13], основанный на произведении Кронекера, с более быстрым построением индекса и меньшим потреблением памяти. Если необходимо извлекать пути много раз для однажды построенного индекса или если изменения в индексе могут быть эффективно подсчитаны динамически, то предлагаемый матричный алгоритм вычисления КС-запросов предпочтительнее.

**Заключение**

В данной статье мы предлагаем новую модификацию матричного алгоритма вычисления КС-запросов для семантики всех путей. Мы реализовали предложенный алгоритм с использованием реализации стандарта GraphBLAS и сравнили полученную реализацию с наиболее эффективными алгоритмами вычисления КС-запросов, основанными на операциях линейной алгебры.

В дальнейшем, предложенному алгоритму необходимы улучшения в потреблении памяти при построении индекса, а для алгоритма, основанного на произведении Кронекера, необходимо ускорение выявления путей.

**Литература**

1. Rehof J., Fähndrich M. 2001. Type-Base Flow Analysis: From Polymorphic Subtyping to CFL-Reachability // SIGPLAN V. 36, N 3, 2001, P. 54–66. doi: 10.1145/373243.360208
2. Zheng X., Rugina R. Demand-driven Alias Analysis for C //In Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’08). ACM, New York, NY, USA, 2008. P. 197–208. doi: 10.1145/1328438.1328464
3. Sevon P., Eronen L. Subgraph Queries by Context-free Grammars // Journal of Integrative Bioinformatics, V. 5, N 2, 2008. P. 157–172. doi: 10.1515/jib-2008-100
4. Zhang X. et al. Context-Free Path Queries on RDF Graphs // In The Semantic Web – ISWC, 2016, Springer International Publishing, Cham, P. 632–648.
5. Medeiros C. et al. Efficient Evaluation of Context-free Path Queries for Graph Databases // In Proceedings of the SAC’18. ACM, New York, NY, USA, 2018. P. 1230–1237. doi: 10.1145/3167132.3167265
6. Santos F. et al. A Bottom-Up Algorithm for Answering Context-Free Path Queries in Graph Databases // In Web Engineering, Springer International Publishing, Cham, 2018. P. 225–233.
7. Grigorev S., Ragozina A. Context-free Path Querying with Structural Representation of Result // In Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR’17), 2017, ACM, New York, NY, USA, Article 10, 7 pages. doi: 10.1145/3166094.3166104
8. Verbitskaia E., Grigorev S., Avdyukhin D. Relaxed Parsing of Regular Approximations of String-Embedded Languages // In Perspectives of System Informatics, Springer International Publishing, Cham, 2016. P. 291–302.
9. Verbitskaia E. et al. Parser Combinators for Context-free Path Querying // In Proceedings of the 9th ACM SIGPLAN International Symposium on Scala (Scala 2018). ACM, New York, NY, USA, 2018. P. 13–23. doi: 10.1145/3241653.3241655
10. Kuijpers J. et al. An Experimental Study of Context-Free Path Query Evaluation Methods // In Proceedings of the SSDBM’19. ACM, New York, NY, USA, 2019. P. 121–132. doi: 10.1145/3335783.3335791
11. Azimov R., Grigorev S. Context-free Path Querying by Matrix Multiplication // In Proceedings of the GRADES-NDA’18. ACM, New York, NY, USA, 2018. Article 5, 10 pages. doi: 10.1145/3210259.3210264
12. Terekhov A. et al. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication // In Proceedings of the GRADES-NDA’20. ACM, New York, NY, USA, 2020. Article 5, 12 pages. doi: 10.1145/3398682.3399163
13. Orachev E. et al. Context-Free Path Querying by Kronecker Product // In European Conference on Advances in Databases and Information Systems. Springer, 2020. P. 49–59.
14. Chomsky N. On certain formal properties of grammars. Information and control, Vol. 2, N 2, 1959. P. 137–167.
15. Abiteboul S. et al. Foundations of Databases: The Logical Level (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA, 1995.

1. Исходный код предложенного алгоритма. [Электронный ресурс]. – Режим доступа: <https://github.com/JetBrains-Research/CFPQ_PyAlgo>, свободный. Яз. англ. (дата обращения 01.04.2021). [↑](#footnote-ref-1)