УНИВЕРСИТЕТ ИТМО

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2021 Том 21 № 4 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
July–August 2021 Vol. 21 No 4 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК УНИВЕРСИТЕТ ИТМО
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2021-21-4-499-505
УДК 004.421.2:519.17 004.657

Алгоритм поиска всех путей в графе с заданными контекстно-свободными ограничениями с использованием матриц с множествами промежуточных вершин

Рустам Шухратуллович Азимов[1]✉, Семён Вячеславович Григорьев[2]

[1] ООО «ИнтеллиДжей Лабс» Ltd., Санкт-Петербург, 197374, Российская Федерация
[1,2] Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация

[1] rustam.azimov19021995@gmail.com✉, https://orcid.org/0000-0003-0223-5172
[2] s.v.grigoriev@spbu.ru, https://orcid.org/0000-0002-7966-0698

Аннотация

Предмет исследования. Рассмотрена задача поиска путей в графе, которые удовлетворяют заданным контекстно-свободным ограничениям. Задача заключается в поиске всех путей в помеченном ориентированном графе, метки на ребрах которых образуют слова из языка, порожденного входной контекстно-свободной грамматикой. Существует два эффективных подхода к решению поставленной задачи с использованием операций линейной алгебры: с помощью обычного матричного умножения для поиска только одного пути и произведения Кронекера для поиска всех путей в графе. В настоящее время не существует алгоритма, который применяет обычное матричное умножение и способен найти все пути в соответствии с заданным контекстно-свободным ограничением. В работе предложен алгоритм поиска всех путей в графе с заданным контекстно-свободным ограничением, который основан на обычном матричном умножении. **Метод.** В матрицу смежности входного графа для каждой пары вершин добавляется дополнительная информация о найденных путях между этими вершинами в виде множества возможных промежуточных вершин. На первом этапе осуществлено построение множества матриц, хранящих в себе информацию обо всех путях, удовлетворяющих заданным ограничениям. На втором этапе выполнено построение искомых путей. **Основные результаты.** Предложенный алгоритм реализован в программе на языке C++. Проведено сравнение с эффективными алгоритмами поиска путей в графе с заданными контекстно-свободными ограничениями. Результаты экспериментального исследования показали, что предложенный алгоритм эффективнее (до 1000 раз быстрее) строит искомые пути, однако в некоторых случаях потребляет до 150 раз больший объем памяти, чем алгоритм, основанный на произведении Кронекера. **Практическая значимость.** Предложенный алгоритм может быть применен 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

поиск путей в графе, 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.

Ссылка для цитирования: Азимов Р.Ш., Григорьев С.В. Алгоритм поиска всех путей в графе с заданными контекстно-свободными ограничениями с использованием матриц с множествами промежуточных вершин // Научно-технический вестник информационных технологий, механики и оптики. 2021. Т. 21, № 4. С. 499–505. doi: 10.17586/2226-1494-2021-21-4-499-505

Context-free path querying with all-path semantics using matrices with sets of intermediate vertices

Rustam Sh. Azimov[1]✉, Semyon V. Grigorev[2]

[1] IntelliJ Labs Co. Ltd., Saint Petersburg, 197374, Russian Federation
[1,2] Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation

[1] rustam.azimov19021995@gmail.com✉, https://orcid.org/0000-0003-0223-5172
[2] s.v.grigoriev@spbu.ru, https://orcid.org/0000-0002-7966-0698

Abstract

The study considers the problem of context-free path querying with all-path query semantics. This problem consists in finding all 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, 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 the algorithm for context-free path query evaluation using the matrix multiplication, which is capable of processing queries with the all-path query semantics. 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. The proposed algorithm was implemented in C++ and a comparison was made with other most efficient algorithms for evaluating context-free path queries, namely with the matrix-based algorithm that allows us to find only one such path, and with the Kronecker product-based algorithm that allows us to find all such paths in the graph. 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. The described algorithm can be applied in 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

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

Acknowledgements

The reported study was funded by RFBR, project No. 19-37-90101.

For citation: Azimov R.Sh., Grigorev S.V. Context-free path querying with all-path semantics using matrices with sets of intermediate vertices. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2021, vol. 21, no. 4, pp. 499–505 (in Russian). doi: 10.17586/2226-1494-2021-21-4-499-505

Введение

В современном мире появляется все больше данных, которые требуют обработки и анализа. При этом графы — одна из самых распространенных и удобных структур данных, позволяющая компактно представлять большие объемы информации и реализовывать эффективные алгоритмы для анализа этой информации. В процессе анализа графа, например, может исследоваться существование определенных путей, достижимость некоторых вершин, и т. д. Графы используются в статическом анализе кода [1, 2], биоинформатике [3] и в сетевом анализе [4]. В настоящее время активно развиваются графовые базы данных (Neo4j, RedisGraph и др.), используемые для хранения и реализации запросов к данным, представленных в виде графов.

Одна из важных задач анализа графов — поиск путей, обладающих заданными свойствами, при этом может решаться, например, задача поиска одного пути или всех путей. Если в результате анализа графа нет необходимости получить такие пути, то решается задача достижимости, в которой исследуется лишь вопрос существования определенных путей.

Для описания свойств искомых путей в помеченном графе возможно задавать ограничения с помощью формальных грамматик над некоторым алфавитом. С помощью таких грамматик ограничивают множество слов, получаемых конкатенацией меток на ребрах рассматриваемых путей. В настоящее время активно исследуются ограничения, представленные в виде контекстно-свободных (КС) грамматик. Этот подход позволяет описывать более широкий набор ограничений, чем, например, используемые на практике, регулярные выражения.

В научных публикациях предложен ряд алгоритмов поиска путей в графе с заданными контекстно-свободными ограничениями, которые основаны на различных методах синтаксического анализа: на основе (G)LL (Generalized Left-to-right, Leftmost derivation) и (G)LR (Generalized Left-to-right, Rightmost derivation in reverse) алгоритмов [5–8]; на основе CYK (Cocke–Younger–Kasami) алгоритма [4]; с использованием парсер-комбинаторов [9]. Исследование Йохема Куиджперса (Jochem Kuijpers) [10] показывает, что существующие решения не применимы для анализа реальных графов из-за значительных времени работы и потребления памяти.

Распространенным способом улучшения производительности алгоритмов анализа графов является их переформулирование в терминах линейной алгебры. В такой формулировке в основном используются операции над матрицами и векторами. Для алгоритмов, которые позволяют найти такую формулировку, возможно применить для представления графов разреженные матрицы (матрицы, имеющие малое количество ненулевых

элементов) и использовать параллельные вычисления, в частности, на основе GPU-технологий. Кроме того, такие алгоритмы зачастую просты в реализации, так как позволяют использовать существующие библиотеки линейной алгебры (GraphBLAS::SuiteSparse, cuSPARSE, cuBLAS, m4ri, Scipy и др.). Такая формулировка была найдена для задач достижимости [11] и поиска одного пути [12] в графе, с заданными контекстно-свободными ограничениями. Алгоритмы решения данных задач основаны на матричных операциях и позволяют получить высокоэффективные реализации с помощью существующих библиотек линейной алгебры. Другой алгоритм поиска путей в графе основан на произведении Кронекера [13]. Данный алгоритм принципиально отличается от алгоритмов, основанных на обычных матричных операциях, тем, что не требует преобразования в нормальную форму входной КС-грамматики, в то время как прочие алгоритмы требуют преобразовывать входную КС-грамматику, например, в нормальную форму Хомского. Кроме того, алгоритм, основанный на произведении Кронекера, в процессе анализа строит более сложные структуры, тем самым позволяя находить все пути, удовлетворяющие контекстно-свободным ограничениям. Но до сих пор не существует алгоритма, использующего обычное матричное произведение, способного находить все пути, удовлетворяющие данным ограничениям. Преимуществом подхода, использующего обычное матричное произведение, по сравнению с подходом, предложенным в [13] и использующим произведение Кронекера, является построение матриц с более явной информацией об искомых путях в графе, что на практике позволит строить искомые пути значительно быстрее.

Таким образом, в данной работе поставлена задача — разработка алгоритма поиска всех путей с заданными контекстно-свободными ограничениями, использующего обычное матричное умножение.

Постановка задачи

Терминал (терминальный символ) — объект, непосредственно присутствующий в словах формального языка, соответствующего формальной грамматике, и имеющий конкретное, неизменяемое значение. В работе использован символ Σ для обозначения конечного алфавита, который состоит из терминальных символов формальной грамматики.

Нетерминал (нетерминальный символ) — объект, обозначающий какую-либо сущность языка (например: формула, арифметическое выражение, команда) и не имеющий конкретного символьного значения. Используем символ N для обозначения конечного множества нетерминальных символов формальной грамматики.

КС-грамматика в слабой нормальной форме Хомского состоит из четырех компонентов $G = (N, \Sigma, P, S)$, где $S \in N$ — стартовый нетерминал и конечные множества: N — нетерминальных и Σ — терминальных символов, P — правил следующего вида:

— $A \to B \cdot C$, для $A, B, C \in N$ и операции конкатенации $(\cdot)$;

— $A \to x$, для $A \in N$, $x \Sigma \cup \{\varepsilon\}$, где ε — пустая строка.

Рассмотрим только КС-грамматики в слабой нормальной форме Хомского, так как для каждой из них можно построить эквивалентную грамматику в данной форме [14]. Операцию конкатенации в правой части правил иногда будем опускать и писать $A \to BC$.

Запишем $A \to {}^*w$, чтобы указать, что строка $w \in \Sigma^*$ может быть получена из нетерминала A некоторой последовательностью применений правил КС-грамматики. А *языком*, задаваемым КС-грамматикой $G = (N, \Sigma, P, S)$, будем называть

$$L(G) = \{w \in \Sigma^* | S \to {}^*w\}.$$

Пусть Σ — конечное множество терминальных символов. *Помеченным графом* назовем ориентированный граф $D = (V, E, \Sigma)$, где V — множество вершин, а $E \subseteq V \times \Sigma \times V$ — множество ребер с метками из алфавита Σ. Для пути π в графе D обозначим $l(\pi)$ — слово, полученное конкатенацией меток на ребрах данного пути. Введем запись $i\pi j$, которая будет обозначать, что в рассматриваемом графе существует путь из вершины $i \in V$ в вершину $j \in V$.

Для заданного графа $D = (V, E)$ и КС-грамматики $G = (N, \Sigma, P, \mathrm{S})$ определим *контекстно-свободное отношение* $R_{G,D} \subseteq V \times V$, являющееся множеством пар вершин, между которыми существует путь, образующий строку, выводимую из стартового нетерминала S. Такое отношение определим следующим образом:

$$R_{G,D} = \{(i, j) | \exists i\pi j (l(\pi) \in L(G))\}.$$

Используя данные определения, сформулируем задачу поиска всех путей в графе с заданными контекстно-свободными ограничениями.

Задача. Проблема поиска всех путей в графе, удовлетворяющих контекстно-свободным ограничениям, для заданного помеченного графа D и КС-грамматики G заключается в вычислении контекстно-свободного отношения $R_{G,D}$, а также для каждой пары вершин $(i, j) \in R_{G,D}$ множества всех путей π между ними, таких что $l(\pi) \in L(G)$.

Множество таких путей может быть бесконечным в случае наличия циклов во входном графе. Обычно для таких множеств используется некоторое конечное представление.

Матричный алгоритм поиска всех путей в графе с заданными контекстно-свободными ограничениями

Предлагаемый алгоритм основан на матричном алгоритме [11], решающем задачу достижимости с заданными контекстно-свободными ограничениями. Этот алгоритм сводит данную задачу к вычислению операций над булевыми матрицами и, как следствие, позволяет использовать высокопроизводительные библиотеки линейной алгебры и современное параллельное оборудование для решения данной задачи анализа графов.

Для каждой пары вершин добавим в ячейки матриц дополнительную информацию о найденных путях между этими вершинами в виде множества возможных промежуточных вершин. Добавленные промежуточные

вершины описывают найденные пути как конкатенации двух меньших путей. Используя эту информацию, возможно восстанавливать все пути, которые образуют слова, выводимые из любого нетерминала заданной КС-грамматики.

Определим матричное умножение $\mathbf{a} \odot \mathbf{b} = \mathbf{c}$, где $\mathbf{a}$ и $\mathbf{b}$ — квадратные матрицы размера $n \times n$, которые имеют множества вершин графа (промежуточных) в качестве элементов, как $\mathbf{c_{i,j}} = \bigcup_{k=1}^{n} (\mathbf{a_{i,k}} \otimes \mathbf{b_{k,j}})$, где

$$\mathbf{a_{i,k}} \otimes \mathbf{b_{k,j}} = \begin{cases} \{k\}, \text{ если } \mathbf{a_{i,k}} \neq \varnothing \text{ и } \mathbf{b_{k,j}} \neq \varnothing \\ \varnothing, \text{ иначе} \end{cases}.$$

Применим операцию «+» поэлементного сложения матриц $\mathbf{a}$ и $\mathbf{b}$ одинакового размера: $\mathbf{a} + \mathbf{b} = \mathbf{c}$, где $\mathbf{c_{i,j}} = \mathbf{a_{i,j}} \cup \mathbf{b_{i,j}}$.

Используя введенные матричные операции, представим матричный алгоритм поиска всех путей в графе с заданными контекстно-свободными ограничениями. Псевдокод данного алгоритма приведен на рис. 1.

Для заданных КС-грамматики $G = (N, \Sigma, P, S)$ и помеченного графа $D = (V, E, \Sigma)$ результат работы представленного алгоритма — набор матриц $\mathbf{T}$ (так называемый индекс), который хранит информацию обо всех путях в графе D, образующих слово, выводимое из некоторого нетерминала КС-грамматики G. В строках 4 и 5 представленного алгоритма добавим специальное значение n к ячейкам $\mathbf{T}^A_{i,j}$, чтобы указать, что это путь с одним ребром или пустой путь π_ε.

Построенный индекс дает возможность вычислить контекстно-свободное отношение $R_{G,D}$ (включая в него все пары вершин (i, j), для которых $\mathbf{T}^S_{i,j} \neq \varnothing$), а также построить любой из найденных путей, и тем самым решить задачу поиска всех путей в графе с заданными контекстно-свободными ограничениями. 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 фиксированное количество путей или ограничить сверху длину путей.

Кроме алгоритма построения индекса, авторы предлагают алгоритм построения найденных путей, псевдокод которого приведен на рис. 2. Если для заданных i, j, A, соответствующий элемент матрицы равен $\varnothing$, то алгоритм возвращает пустое множество, так как путей искомого вида не существует. Данный алгоритм возвращает множество с пустым путем π_ε, только если $i = j$ и $A \to \varepsilon \in P$. В строке 20 использована операция $(\cdot)$, которая естественным образом обобщает операцию конкатенации путей, строя все возможные конкатенации пар путей из двух множеств. Предположим, что множества путей вычисляются «лениво», чтобы была обеспечена завершаемость алгоритма в случае бесконечного количества путей.

```
1: function AllPathCFPQ(
       D = (V, E, Σ),                          ▷ Входной помеченный граф
       G = (N, Σ, P, S))                       ▷ Входная КС-грамматика
2:     n ← |V|
3:     T ← {T^A | A ∈ N, T^A − матрица n × n, T^A_{i,j} ← ∅ }
4:     for all (i, x, j) ∈ E, A | A → x ∈ P do T^A_{i,j} ← {n}
5:     for all A | A → ε ∈ P do T^A_{i,i} ← {n}
6:     while любая матрица из T изменяется do
7:         for all A → BC ∈ P, где T^B или T^C изменились do
8:             T^A ← T^A + (T^B ⊙ T^C)
9:     return T
```

Рис. 1. Матричный алгоритм поиска всех путей в графе с заданными контекстно-свободными ограничениями

Fig. 1. A matrix-based context-free path querying algorithm with all-path query semantics

Эксперименты

Цель экспериментального исследования — изучение применимости предложенного матричного алгоритма и сравнение с лучшими реализациями алгоритмов поиска путей в графе с заданными контекстно-свободными ограничениями, использующими операции линейной алгебры, а именно:

— *MtxSingle* — реализация матричного алгоритма поиска одного пути в графе с заданными контекстно-свободными ограничениями [12];
— *Tns* — реализация алгоритма поиска всех путей в графе с заданными контекстно-свободными ограничениями, основанного на произведении Кронекера [13];
— *MtxAll* — реализация предложенного матричного алгоритма поиска всех путей в графе с заданными контекстно-свободными ограничениями, которая для матричных операций использует реализацию стандарта GraphBLAS.

```
1: function ExtractAllPaths(i, j, A, T = {T^{A_k} | A_k ∈ N}, G = (N, Σ, P, S))
2:     if T^A_{i,j} = ∅ then
3:         return ∅                              ▷ Таких путей не существует
4:     n ← размер квадратной матрицы T^A
5:     resultPaths ← ∅
6:     for all k ∈ T^A_{i,j} do
7:         if k = n then          ▷ Добавляем путь из одного ребра или пустой путь
8:             for all x | A → x ∈ P do
9:                 if (i, x, j) ∈ E then
10:                    resultPaths ← resultPaths ∪ {((i, x, j))}
11:            if (i = j) ∧ (A → ε ∈ P) then
12:                resultPaths ← resultPaths ∪ {π_ε}
13:        else▷ Добавляем конкатенацию путей из i в k и путей из k в j
14:            for all A → BC ∈ P do
15:                index_B ← T^B_{i,k}
16:                index_C ← T^C_{k,j}
17:                if (index_B ≠ ⊥) ∧ (index_C ≠ ⊥) then
18:                    lPaths ← ExtractAllPaths(i, k, B, T, G)
19:                    rPaths ← ExtractAllPaths(k, j, C, T, G)
20:                    resultPaths ← resultPaths ∪ lPaths · rPaths
21:    return resultPaths
```

Рис. 2. Алгоритм построения всех путей, удовлетворяющих заданным контекстно-свободным ограничениям

Fig. 2. A paths extraction algorithm for the context-free path querying

Все реализации используют для вычислений центральный процессор и матрицы в разреженном формате. Выполним измерение времени проведения и размер требуемой памяти для создания индекса. Сравним практическую применимость построения путей для обеих реализаций *MtxAll* и *Tns* алгоритмов поиска всех путей в графе с заданными контекстно-свободными ограничениями. Исходный код всех трех реализаций доступен на GitHub[1]. Для экспериментов выбран персональный компьютер с установленной операционной системой Ubuntu 18.04, процессором Intel Core i7-6700, 3,4 ГГц и оперативной памятью DDR4 64 ГБ.

Данные. Используем графы и соответствующие КС-грамматики из набора данных, представленного в [12], который содержит графы для реальных данных в формате RDF (*pathways, go-hierarchy, enzyme, eclass_514en, go, geospecies, taxonomy*) и КС-грамматики *g1*, *g2*, которые используются для поиска всех вершин в графе, находящихся на одном уровне иерархии [15]. Описанное свойство — один из примеров, который может быть выражен с помощью КС-грамматик, но не с помощью регулярных.

Результаты. Результаты создания индекса для трех реализаций представлены в таблице. Можно увидеть, что матричный алгоритм поиска одного пути позволяет строить индекс быстрее, особенно при работе с большими графами. *MtxSingle* не может построить все найденные при анализе графа пути, так как использует более простой индекс, чтобы восстановить только один путь для каждой пары вершин. Реализация *Tns* алгоритма, основанного на произведении Кронекера, использует более сложный, но компактный индекс, что позволяет не только построить все найденные пути, но и потреблять меньше памяти, чем другие матричные алгоритмы.

Реализация *MtxAll* предлагаемого матричного алгоритма поиска всех путей в графе с заданными контекстно-свободными ограничениями 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 информацию обо всех найденных путях в более явном виде. На самом большом графе *taxonomy* и КС-грамматике *g1* произошла ошибка нехватки памяти для реализации *MtxAll*, что отмечено в таблице с помощью «*error*».

После построения индекса сравним время построения путей для двух реализаций алгоритмов поиска всех путей в графе с заданными контекстно-свободными ограничениями. Результаты построения путей с длиной, не превосходящей 10, для графов *go* и *eclass_514en* представлены на рис. 3 и 4 (используются стандартные диаграммы размаха, желтыми линиями указаны медианы, вертикальными линиями обозначены стандартные интервалы между 5%-ми и 95%-ми перцентилями, выбросы опускаются). После построения путей для каждой пары вершин, сгруппируем время построения по количеству возвращаемых путей. Видно, что время извлечения путей в реализации *MtxAll* предложенного матричного алгоритма до 1000 раз меньше, чем в реализации *Tns*. В предложенном матричном алгоритме строится индекс с более явной информацией о найденных путях, что позволяет значительно быстрее их восстанавливать.

Выводы. По результатам экспериментального исследования можно сделать следующие выводы.

Для задачи поиска одного пути, удовлетворяющего контекстно-свободным ограничениям, матричный алгоритм [12] является наиболее производительным, а алгоритм [13], основанный на произведении Кронекера, потребляет наименьшее количество памяти.

Для задачи поиска всех путей с заданными контекстно-свободными ограничениями, в случае, когда необходимо часто пересчитывать индекс для изменяющегося графа или КС-грамматики, лучшим выбором является алгоритм, основанный на произведении Кронекера, с более быстрым построением индекса и меньшим потреблением памяти. Если необходимо строить пути много раз для однажды построенного индекса или если изменения в индексе могут быть эффективно подсчитаны

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

Таблица. Построение индекса для КС-грамматик *g1*/*g2*

Table. Index construction for the context-free grammars *g1*/*g2*

Граф	|*V*|	|*E*|	КС-грамматики *g1*/*g2*					
			MtxAll		*Tns*		*MtxSingle*	
			Время, с	Память, МБ	Время, с	Память, МБ	Время, с	Память, МБ
pathways	6238	18 598	0,04/0,01	91/49	0,02/0,01	123/122	0,01/0,01	671/671
go-hierarchy	45 007	980 218	22,12/15,66	38 797/28 447	0,17/0,24	265/252	1,41/0,84	660/671
enzyme	48 815	109 695	0,40/0,02	307/61	0,04/0,02	137/132	0,01/0,01	216/217
eclass_514en	239 111	523 727	25,02/0,22	14 416/126	0,24/0,27	205/193	0,23/0,16	216/216
go	272 770	534 311	11,80/1,13	8290/990	1,58/1,27	282/243	1,45/0,93	215/217
geospecies	450 609	2 311 461	4,45/0,34	2691/156	0,08/0,01	218/196	0,06/0,01	2250/2251
taxonomy	5 728 398	14 922 125	*error*/19,13	*error*/27 232	4,42/3,56	2018/1776	2,73/1,15	1962/2250

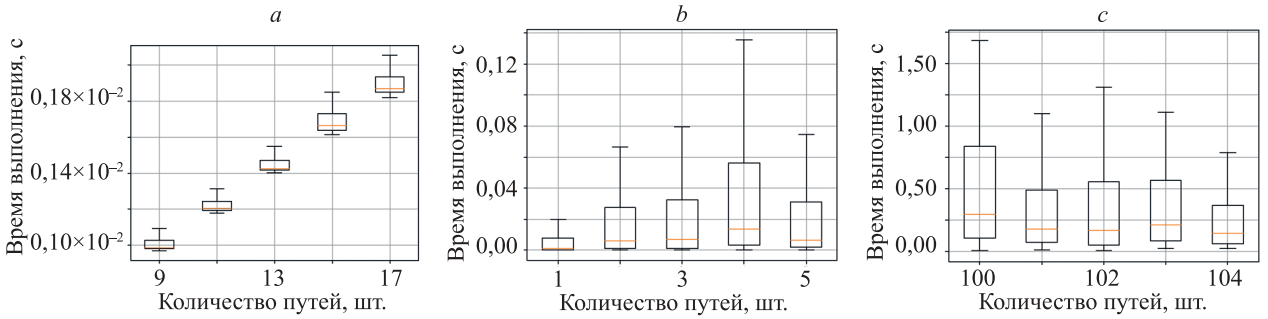

Рис. 3. Время построения путей для реализации *Tns* алгоритма, основанного на произведении Кронекера для грамматики *g1*: для графа *eclass_514en* (*a*); для графа *go* и малого количества путей (*b*); для графа *go* и большого количества путей (*c*)

Fig. 3. Paths construction time for the implementation *Tns* of the Kronecker product-based algorithm using *g1* grammar: for the graph *eclass_514en* (*a*); for the graph *go* and a small number of paths (*b*); for the graph *go* and a large number of paths (*c*)

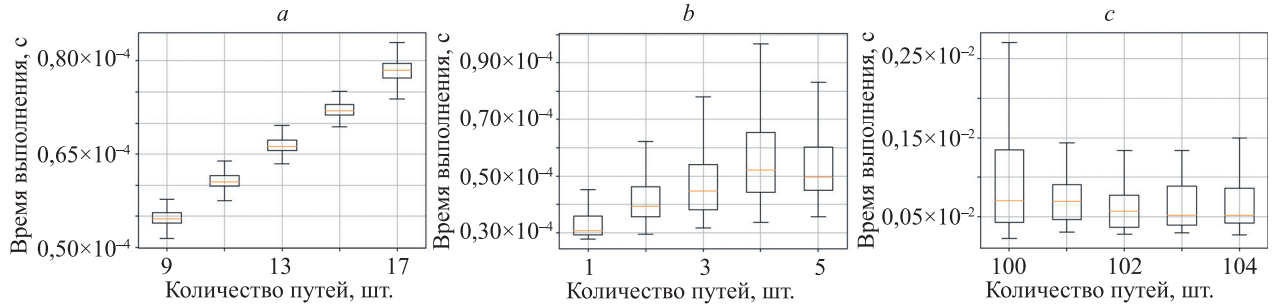

Рис. 4. Время построения путей для реализации *MtxAll* предложенного матричного алгоритма для грамматики *g1*: для графа *eclass_514en* (a); для графа *go* и малого количества путей (*b*); для графа *go* и большого количества путей (*c*)

Fig. 4. Paths construction time for the implementation *MtxAll* of the proposed matrix-based algorithm using *g1* grammar: for the graph *eclass_514en* (*a*); for the graph *go* and a small number of paths (*b*); for the graph *go* and a large number of paths (*c*)

динамически, то предлагаемый матричный алгоритм поиска всех путей с заданными контекстно-свободными ограничениями предпочтительнее.

Заключение

В работе предложен алгоритм поиска всех путей с заданными контекстно-свободными ограничениями, основанный на матричном алгоритме для задачи достижимости.

Алгоритм реализован с использованием реализации стандарта GraphBLAS, что позволяет эффективно анализировать большие графы с применением параллельных вычислений и разреженных форматов для хранения матриц. Выполнено сравнение полученной реализации с реализациями алгоритмов поиска путей в графе с заданными контекстно-свободными ограничениями, основанными на операциях линейной алгебры. В то время как для задачи поиска всех путей, удовлетворяющих контекстно-свободным ограничениям, предложенный алгоритм является наиболее производительным в случае, когда необходимо извлекать пути много раз для однажды построенного индекса или если изменения в индексе могут быть эффективно подсчитаны динамически.

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

Литература

1. Rehof J., Fähndrich M. Type-base flow analysis: from polymorphic subtyping to CFL-reachability // SIGPLAN Notices. 2001. V. 36. N 3. P. 54–66. https://doi.org/10.1145/373243.360208
2. Zheng X., Rugina R. Demand-driven alias analysis for C // Proc. 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08). 2008. P. 197–208. https://doi.org/10.1145/1328438.1328464
3. Sevon P., Eronen L. Subgraph queries by context-free grammars // Journal of Integrative Bioinformatics. 2008. V. 5. N 2. P. 157–172. https://doi.org/10.1515/jib-2008-100
4. Zhang X., Feng Z., Wang X., Rao G.Z., Wu W.R. Context-free path queries on RDF graphs // Lecture Notes in Computer Science. 2016. V. 9981. P. 632–648. https://doi.org/10.1007/978-3-319-46523-4_38

References

1. Rehof J., Fähndrich M. Type-base flow analysis: from polymorphic subtyping to CFL-reachability. *SIGPLAN Notices*, 2001, vol. 36, no. 3, pp. 54–66. https://doi.org/10.1145/373243.360208
2. Zheng X., Rugina R. Demand-driven alias analysis for C. *Proc. 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, 2008, pp. 197–208. https://doi.org/10.1145/1328438.1328464
3. Sevon P., Eronen L. Subgraph queries by context-free grammars. *Journal of Integrative Bioinformatics*, 2008, vol. 5, no. 2, pp. 157–172. https://doi.org/10.1515/jib-2008-100
4. Zhang X., Feng Z., Wang X., Rao G.Z., Wu W.R. Context-free path queries on RDF graphs. *Lecture Notes in Computer Science*, 2016, vol. 9981, pp. 632–648. https://doi.org/10.1007/978-3-319-46523-4_38

5. Medeiros C., Musicante M.A., Costa U.S. Efficient evaluation of context-free path queries for graph databases // Proc. 33rd Annual ACM Symposium on Applied Computing (SAC-2018). 2018. P. 1230–1237. https://doi.org/10.1145/3167132.3167265
6. Santos F., Costa U.S., Musicante M.A. A bottom-up algorithm for answering context-free path queries in graph databases // Lecture Notes in Computer Science. 2018. V. 10845. P. 225–233. https://doi.org/10.1007/978-3-319-91662-0_17
7. Grigorev S., Ragozina A. Context-free path querying with structural representation of result // Proc. 13th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR'17). 2017. P. 3166104. https://doi.org/10.1145/3166094.3166104
8. Verbitskaia E., Grigorev S., Avdyukhin D. Relaxed parsing of regular approximations of string-embedded languages // Lecture Notes in Computer Science. 2016. V. 9609. P. 291–302. https://doi.org/10.1007/978-3-319-41579-6_22
9. Verbitskaia E., Kirillov I., Nozkin I., Grigorev S. Parser combinators for context-free path querying // Proc. 9th ACM SIGPLAN International Symposium on Scala (Scala 2018). 2018. P. 13–23. https://doi.org/10.1145/3241653.3241655
10. Kuijpers J., Fletcher G., Yakovets N., Lindaaker T. An experimental study of context-free path query evaluation methods // Proc. 31st International Conference on Scientific and Statistical Database Management (SSDBM-2019). 2019. P. 121–132. https://doi.org/10.1145/3335783.3335791
11. Azimov R., Grigorev S. Context-free path querying by matrix multiplication // Proc. 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems and Network Data Analytics (GRADES-NDA 2018). 2018. P. a5. https://doi.org/10.1145/3210259.3210264
12. Terekhov A., Khoroshev A., Azimov R., Grigorev S. Context-free path querying with single-path semantics by matrix multiplication // Proc. 3rd ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems and Network Data Analytics (GRADES-NDA 2020). 2020. P. 5. https://doi.org/10.1145/3398682.3399163
13. Orachev E., Epelbaum I., Azimov R., Grigorev S. Context-free path querying by kronecker product // Lecture Notes in Computer Science. 2020. V. 12245. P. 49–59. https://doi.org/10.1007/978-3-030-54832-2_6
14. Chomsky N. On certain formal properties of grammars // Information and Control. 1959. V. 2. N 2. P. 137–167. https://doi.org/10.1016/S0019-9958(59)90362-6
15. Abiteboul S., Hull R., Vianu V. Foundations of Databases: The Logical Level. 1st ed. Addison-Wesley, 1995. 704 p.

5. Medeiros C., Musicante M.A., Costa U.S. Efficient evaluation of context-free path queries for graph databases. *Proc. 33rd Annual ACM Symposium on Applied Computing (SAC-2018)*, 2018, pp. 1230–1237. https://doi.org/10.1145/3167132.3167265
6. Santos F., Costa U.S., Musicante M.A. A bottom-up algorithm for answering context-free path queries in graph databases. *Lecture Notes in Computer Science*, 2018, vol. 10845, pp. 225–233. https://doi.org/10.1007/978-3-319-91662-0_17
7. Grigorev S., Ragozina A. Context-free path querying with structural representation of result. *Proc. 13th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR'17)*, 2017, pp. 3166104. https://doi.org/10.1145/3166094.3166104
8. Verbitskaia E., Grigorev S., Avdyukhin D. Relaxed parsing of regular approximations of string-embedded languages. *Lecture Notes in Computer Science*, 2016, vol. 9609, pp. 291–302. https://doi.org/10.1007/978-3-319-41579-6_22
9. Verbitskaia E., Kirillov I., Nozkin I., Grigorev S. Parser combinators for context-free path querying. *Proc. 9th ACM SIGPLAN International Symposium on Scala (Scala 2018)*, 2018, pp. 13–23. https://doi.org/10.1145/3241653.3241655
10. Kuijpers J., Fletcher G., Yakovets N., Lindaaker T. An experimental study of context-free path query evaluation methods. *Proc. 31st International Conference on Scientific and Statistical Database Management (SSDBM-2019)*, 2019, pp. 121–132. https://doi.org/10.1145/3335783.3335791
11. Azimov R., Grigorev S. Context-free path querying by matrix multiplication. *Proc. 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems and Network Data Analytics (GRADES-NDA 2018)*, 2018, pp. a5. https://doi.org/10.1145/3210259.3210264
12. Terekhov A., Khoroshev A., Azimov R., Grigorev S. Context-free path querying with single-path semantics by matrix multiplication. *Proc. 3rd ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems and Network Data Analytics (GRADES-NDA 2020)*, 2020, pp. 5. https://doi.org/10.1145/3398682.3399163
13. Orachev E., Epelbaum I., Azimov R., Grigorev S. Context-free path querying by kronecker product. *Lecture Notes in Computer Science*, 2020, vol. 12245, pp. 49–59. https://doi.org/10.1007/978-3-030-54832-2_6
14. Chomsky N. On certain formal properties of grammars. *Information and Control*, 1959, vol. 2, no. 2, pp. 137–167. https://doi.org/10.1016/S0019-9958(59)90362-6
15. Abiteboul S., Hull R., Vianu V. *Foundations of Databases: The Logical Level*. 1st ed. Addison-Wesley, 1995, 704 p.

Авторы

Азимов Рустам Шухратуллович — программист, ООО «ИнтеллиДжей Лабс» Ltd., Санкт-Петербург, 197374, Российская Федерация; ассистент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, SC 57203022098, https://orcid.org/0000-0003-0223-5172, rustam.azimov19021995@gmail.com

Григорьев Семён Вячеславович — кандидат физико-математических наук, доцент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, SC 56047575300, https://orcid.org/0000-0002-7966-0698, s.v.grigoriev@spbu.ru

Статья поступила в редакцию 23.04.2021
Одобрена после рецензирования 17.05.2021
Принята к печати 25.07.2021

Authors

Rustam Sh. Azimov — Software Developer, IntelliJ Labs Co. Ltd., Saint Petersburg, 197374, Russian Federation; Assistant, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, SC 57203022098, https://orcid.org/0000-0003-0223-5172, rustam.azimov19021995@gmail.com

Semyon V. Grigorev — PhD, Associate Professor, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, SC 56047575300, https://orcid.org/0000-0002-7966-0698, s.v.grigoriev@spbu.ru

Received 23.04.2021
Approved after reviewing 17.05.2021
Accepted 25.07.2021