ContextFree Wars: The RedisGraph Strikes Back

Arseniy Terekhov
simpletondl@yandex.ru
Saint Petersburg State University
St. Petersburg, Russia

Vlada Pogozhelskaya
pogozhelskaya@gmail.com
Saint Petersburg State University
St. Petersburg, Russia

Vadim Abzalov
!!!@!!!
Saint Petersburg State University
St. Petersburg, Russia

Timur Zinnatulin
!!!@!!!
Saint Petersburg State University
St. Petersburg, Russia

Semyon Grigorev
s.v.grigoriev@spbu.ru
semyon.grigorev@jetbrains.com
Saint Petersburg State University
St. Petersburg, Russia
JetBrains Research
St. Petersburg, Russia

Figure 1: Episode IV: A New Hope

ABSTRACT

A long time ago in a galaxy far far away...

1 INTRODUCTION

Language-constrained path querying [2] is a way to find paths in edge-labeled graphs when constraints are formulated in terms of language which restrict words formed by paths: the word formed by path's labels concatenation should be in the specified language. This way is very natural for navigational queries in graph databases, and one of the most popular languages which are used for constraints is a regular language. But in some cases, regular languages are not expressive enough, as a result, context-free languages gain popularity. Constraints in the form of context-free languages, or context-free path querying (CFPQ), can be used for RDF analysis [10], biological data analysis [8], static code analysis [7, 11], and in other areas.

Big amount of research done on CFPQ, a number of CFPQ algorithms were proposed, but the application of context-free constraints for real-world data analysis faced with some problems problem. The first problem is a bad performance of proposed algorithms on real-world data, as was shown by Jochem Kuijpers et al. [4]. Moreover, there are no graph databases with full-stack support of CFPQ, the main effort was made in algorithms and their theoretical properties research. This fact hinders research of problems reducible to CFPQ, thus it hinders the development of new solutions for some problems. For example, recently graph segmentation in data provenance analysis was reduced to CFPQ [5], but authors faced the problem during the evaluation of the proposed approach: no one graph database support CFPQ.

In [1] Rustam Azimov propose a matrix-based algorithm for CFPQ. This algoritm is one of promissing way to solve the first problem and provide appropriate solution for real-world data analysis, as was shown by Nikita Mishim et al. in [6] and Arseniy Terekhov et al. in [9]. But this algorithm always computes information (reachability facts or single path which satisfies constraints) for all pairs of vertices in the graph, namely it solves *all-pairs* problem. It is unreasonable for some real-world scenarios when one can provide a relatively small set of start vertices or even single start vertex.

While all-pairs context-free path querying is a classical problem that investigates in a number of works, there is no, in our knowledge, solutions for single-source and multiple-source CFPQ. In this work we propose a matrix-based *multiple-source* (and *single-source* as a partial case) CFPQ algorithm.

Also, we provide full-stack support of CFPQ for the RedisGraph[1] [3] graph database. We implement a Cypher query language extension[2] that allows one to express context-free constraints, and extend the RedisGraph to support this extension. In our knowledge, it is the first full-stack implementation of CFPQ.

To summarize, we make the following contribution in this paper.

(1) We modify Azimov's matrix-based CFPQ algorithm and provide a multiple-source matrix-based CFPQ algorithm. As a partial case, it is possible to use our algorithm in a single-source scenario. Our modification still based on

[1] RedisGraph graph database Web-page: https://redislabs.com/redis-enterprise/redis-graph/. Access date: 19.07.2020.
[2] Proposal which describes path patterns specification syntax for Cypher query language: https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc. The proposed syntax allows one to specify context-free constraints. Access date: 19.07.2020.

linear algebra, hance it is simple to implementation and allows one to use high-performance libraries for implmementation.
(2) We evalute the proposed algorithm. Our evaluation shows that !!!
(3) We provide full-stack support of CFPQ by extending the RedisGraph graph database. To do it, we extend Cypher with syntax allows one to express context-free constraints, implement the proposed algorithm in a RedisGraph backend, and support new syntax in the RedisGraph query execution engine. Finally, evaluate the poposed solution.

2 PRELIMINARIES

In this section we introduce common definitions in graph theory and formal language theory which will be used in this paper. Also, we provide brief description of Azimov's algorithm which is used as a base of our solution.

2.1 Graphs

In this work we use edge-labelled digraph as a data model and define it as follows.

Definition 2.1. Edge-labelled Digraph

An example of the graph is presented in figure 2.

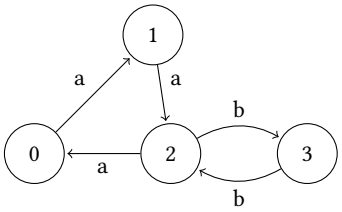

Figure 2: The example of input graph $\mathcal{G}$

We use adjacency matrix decomposed to a set of a boolean matrix as a representation of the graph.

Definition 2.2. An adjacency matrix M of the graph $\mathcal{G} =$ is a square $|V| \times |V|$ matrix, such that $M[i,j] = \{l \mid e = (i,l,j) \in E\}$.

Adjacency matrix M of the graph $\mathcal{G}$ is

$$M = \begin{pmatrix} . & \{a\} & . & . \\ . & . & \{a\} & . \\ \{a\} & . & . & \{b\} \\ . & . & \{b\} & . \end{pmatrix}.$$

Definition 2.3. Boolean decomposition of adjacency matrix M of graph $\mathcal{G} =$ is set of Boolean matrix

$$\mathcal{M} = \{M^l \mid l \in L, M^l[i,j] = 1 \iff l \in M[i,j]\}.$$

Matrix M can be represented as a set of two Boolean matrices M^a and M^b where

$$M^a = \begin{pmatrix} . & 1 & . & . \\ . & . & 1 & . \\ 1 & . & . & . \\ . & . & . & . \end{pmatrix}, M^b = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & 1 \\ . & . & 1 & . \end{pmatrix} \quad (1)$$

2.2 Languages

Grammars, normal forms, ...

Definition 2.4. Grammar

Definition 2.5. Language

2.3 Matrix-Based Algorithm

Description

Pseudocode.

Comments and conclusion

3 MATRIX-BASED MULTIPLE-SOURCE CFPQ ALGORITHM

In this section we introduce two versions of multiple-source matrix-based CFPQ algorithm. This algorithm is a modification of Azimov's matrix-based algorithm for CFPQ and the idea is that we cut off those vertices from which we are not interested in paths.

Let $D = (V, E)$ be the input graph, $G = (N, \sigma, P, S)$ be the input grammar and Src be the input set of vertices. We want to find all $(i, j) : \exists i \pi j$ and $i \in Src$.

Listing 1 Context-free path querying algorithm

```
1:  function MultiSrcCFPQ(D = (V, E), G = (N, Σ, P, S), Src)
2:      T ← {T^A | A ∈ N, T^A ← ∅}        ▷ Matrix in which every element is ∅
3:      TSrc ← {TSrc^A | A ∈ N \ S, TSrc^A ← ∅}   ▷ Matrix for input vertices in which every element is ∅
4:      for all v ∈ Src do                 ▷ Input matrix initialization
5:          TSrc^S_{v,v} ← true
6:      for all A → x ∈ P do               ▷ Simple rules initialization
7:          for all (v, x, to) ∈ E do
8:              T^A_{v,to} ← true
9:      while T or TSrc is changing do     ▷ Algorithm's body
10:         for all A → BC ∈ P do
11:             M ← TSrc^A * T^B
12:             T^A ← T^A + M * T^C
13:             TSrc^B ← TSrc^B + TSrc^A
14:             TSrc^C ← TSrc^C + getDst(M)
15:     return T
16:
17: function getDst(M)
18:     A ← ∅
19:     for all (v, to) ∈ V^2 | M_{v,to} = true do
20:         A_{to,to} ← true
21:     return A
```

The operation of transitive closure calculation from Azimov's algorithm is supplemented with one more matrix multiplication which saves only vertices we are interested in.

We also provide one more version of algorithm which has memory optimization. It is noticed that every time we want to find all paths from the certain set of vertices, the first version of algorithm calculates everything from scratch. Since recalculating might take the significant amount of time, the second version is specified for such scenarios. This version stores all the vertices, the paths from which have already been calculated.

3.1 Implementation Details

Both versions of algorithm [3] are based on the GraphBLAS framework that allows you to represent graphs as matrices and work with them in terms of linear algebra. For convenience, all the code is written in Python using pygraphblas, which is Python wrapper

[3] https://github.com/JetBrains-Research/CFPQ_PyAlgo

Listing 2 Context-free path querying algorithm

```
1: function MultiSrcCFPQSmart(index = (D, G, T, TSrc), Src)
2:     TNewSrc ← {TNewSrc^A | A ∈ N \ S, TNewSrc^A ← ∅}
3:     for all v ∈ Src | index.TSrc_{v,v} = false do
4:         TNewSrc^S_{v,v} ← true
5:     while index.T or TNewSrc is changing do
6:         for all A → BC ∈ P do
7:             M ← TNewSrc^A * index.T^B
8:             index.T^A ← index.T^A + M * index.T^C
9:             TNewSrc^B ← TNewSrc^B + TNewSrc^A \ index.TSrc^B
10:            TNewSrc^C ← TNewSrc^C + getDst(M) \ index.TSrc^C
```

around GraphBLAS API and based on SuiteSparse:GraphBLAS — the full implementation of GraphBLAS standart.

3.2 Algorithm Evaluation

And comparison. With combinators, GLL (.NET version).

Evaluation setup. Hardware basic description.

Graphs and queries from CFPQ_Data[4] Graphs and queries description: #V, #E, types of queries.

Tables.

Graphics (boxes). 1,2,4,8,16,32,50,100,500,1000,5000

Results.

Conclusion.

4 CFPQ FULL-STACK SUPPORT

In order to provide full-stack support of CFPQ it is necessatry to choose an appropriatr graph database. It was shown by Arseniy Terekhov et al. in [9] that matrix-based algorithm can be naturelly integrated into RedisGraph graph database because both, the algorithm and the database, operates over matrix representation of graphs. Moreover, RedisGraph supports Cypher as a query language and there is a proposal which describes Cypher expetsion which allows one to specify context-free constraints. Thus we choose RedisGraph as a base for our solution.

4.1 Cypher Extending

The first what we should do is to extend Cypher to be able to express context-free constraints. There is a description of the respective Cypher syntax extension[5], proposed by Tobias Lindaaker, but this syntax does not implement yet in Cypher parsers.

RedisGraph database supports subset of Cypher language and useses `libcypher-parser`[6] library to parse queries. We extend this library by intoducing new syntax proposed !!! We implement[7] full extension, not only part which is necessary for simple CFPQ.

Main feature which allows one to specify cintext-free constraints is a *named path patterns*: one can specify a name for pattern and after that use it in other patterns, or in the same pattern. Using this feature, structure of query is pretty similar to context-free grammar. For exmaple !!!

Examples of quries. Description o examples.

4.2 RedisGraph Extending

CFPQ to matrix expressions, etc. General schema of integration.

Limits, restrictions, examples, etc.

4.3 Evaluation

Small basic evalustion on real-world graph (geo?). In order to show, that performance is reasonable.

Regular quries. Comparison with other DB?

5 CONCLUSION

In this paper we propose a number of multiple-source modifications of Azimov's CFPQ algorithm. Evaluation of the proposed modifications on the real-world examples shows that !!!! Finally, we provide the full-stack support of CFPQ. For our solution we implement corresponding Cypher extension as a part of `libcypher-parser`, integrate the proposed algorithm into RedisGraph, and extend RedisGraph execution plan builder to support extended Cypher queres. We demonstrate, that our solution allows one evaluate not only context-free queryes, but also regular one.

In the future, it is necessary to provide formal translation of Cypher to linear algebra, or find a maximal subset of Cypher which can be translated to linear algebra. There is a number of work on a subset of SPARQL to linear algebra translation, such as [**?**], but they are very limited. Deep investigation of this topic helps one to realize limits and restrictions of linear algebra utilization for graph databases. Moreover, it helps to improve existing solutions.

We show that evaluation of regular queryes is possible in practice by using CFPQ algorithm, as far as regular queries is a partial cas of the context-free one. But it seems, that the proposed solution is not optimal. For real-world solutions it is important to provide an optimal unified algorithm for both RPQ and CFPQ. One of possible way to solve this provlem is to use tensor-based algorithm [**?**].

Another imprtant task is to compare non-linear-algebra-based approaches to multiple-source CFPQ with the proposed solution. In [**?**] Johem Kujpers et.al. shows that all-pairs CFPQ algorithms implemented in Neo4j demonstrate unreasonable performance on real-world data for Neo4j. At the same time, Arseniy Terekhov et.al. shows that matrix-based all-pairs CFPQ algortihm implemented in appropriate linear algebra based graph database (RedisGraph) demonstrates good performance. But in the case of multiple-source scenario, when a number of sources is relatively small, non-linear-algebra-based solutions can be better, because such solutions naturally handle small reqired subgraph.

[4] !!!

[5] Formal syntax specification: https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc#11-syntax. Access date: 19.07.2020.

[6] The `libcypher-parser` is an open-source parser library for Cypher query language. GitHub repository of the project: https://github.com/cleishm/libcypher-parser. Access date: 19.07.2020.

[7] The modified libsypher-pareser library with support of syntax for path patterns: https://github.com/YaccConstructor/libcypher-parser. Access date: 19.07.2020.

REFERENCES

[1] Rustam Azimov and Semyon Grigorev. 2018. Context-free Path Querying by Matrix Multiplication. In *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) (GRADES-NDA '18).* ACM, New York, NY, USA, Article 5, 10 pages. https://doi.org/10.1145/3210259.3210264

[2] C. Barrett, R. Jacob, and M. Marathe. 2000. Formal-Language-Constrained Path Problems. *SIAM J. Comput.* 30, 3 (2000), 809–837. https://doi.org/10.1137/S0097539798337716 arXiv:https://doi.org/10.1137/S0097539798337716

[3] P. Cailliau, T. Davis, V. Gadepally, J. Kepner, R. Lipman, J. Lovitz, and K. Ouaknine. 2019. RedisGraph GraphBLAS Enabled Graph Database. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).* 285–286.

[4] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management (SSDBM '19)*. ACM, New York, NY, USA, 121–132. https://doi.org/10.1145/3335783.3335791

[5] H. Miao and A. Deshpande. 2019. Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1710–1713.

[6] Nikita Mishin, Iaroslav Sokolov, Egor Spirin, Vladimir Kutuev, Egor Nemchinov, Sergey Gorbatyuk, and Semyon Grigorev. 2019. Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication. In *Proceedings of the 2Nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) (GRADES-NDA'19)*. ACM, New York, NY, USA, Article 12, 5 pages. https://doi.org/10.1145/3327964.3328503

[7] Jakob Rehof and Manuel Fähndrich. 2001. Type-Base Flow Analysis: From Polymorphic Subtyping to CFL-Reachability. *SIGPLAN Not.* 36, 3 (Jan. 2001), 54–66. https://doi.org/10.1145/373243.360208

[8] Petteri Sevon and Lauri Eronen. 2008. Subgraph Queries by Context-free Grammars. *Journal of Integrative Bioinformatics* 5, 2 (2008), 157 – 172. https://doi.org/10.1515/jib-2008-100

[9] Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, and Semyon Grigorev. 2020. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication. In *Proceedings of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) (GRADES-NDA'20)*. Association for Computing Machinery, New York, NY, USA, Article 5, 12 pages. https://doi.org/10.1145/3398682.3399163

[10] Xiaowang Zhang, Zhiyong Feng, Xin Wang, Guozheng Rao, and Wenrui Wu. 2016. Context-Free Path Queries on RDF Graphs. In *The Semantic Web – ISWC 2016*, Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil (Eds.). Springer International Publishing, Cham, 632–648.

[11] Xin Zheng and Radu Rugina. 2008. 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, 197–208. https://doi.org/10.1145/1328438.1328464