

ContextFree Wars: The RedisGraph Strikes Back

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

Vlada Poguzelskaya
!!!@!!!
Saint Petersburg State University
St. Petersburg, Russia

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

Timur Zinnatuln
!!!@!!!
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 [?] 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 [?], biological data analysis [?], static code analysis [?], 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. [?]. 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 [?], but authors faced the problem during the

evaluation of the proposed approach: no one graph database support CFPQ.

In [?] Rustam Azimov propose a matrix-based algorithm for CFPQ. This algorithm is one of promising way to solve the first problem and provide appropriate solution for real-world data analysis, as was shown by Nikita Mishim et al. in [?] and Arseniy Terekhov et al. in [?]. 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 RedisGraph¹ [?] graph database. We implement a Cypher query language extension² 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

¹RedisGraph graph database Web-page: <https://redislabs.com/redis-enterprise/redis-graph/>. Access date: 19.07.2020.

²Proposal 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, hence it is simple to implementation and allows one to use high-performance libraries for implementation.

- (2) We evaluate 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 defin it as follows.

Definition 2.1. Edge-labelled Digraph

An example of the graph is presented in figure ??.

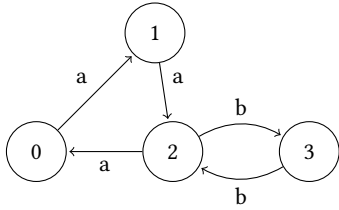


Figure 2: The example of input graph \mathcal{G}

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

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

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

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

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

New algo description.

Code, its description.

Versions, differences between them.

Listing 1 Context-free path querying algorithm

```

1: function MULTISrcCFPQ( $D = (V, E), G = (N, \Sigma, P, S), Src$ )
2:    $T \leftarrow \{T^A \mid A \in N, T^A \leftarrow \emptyset\}$ 
3:    $TSrc \leftarrow \{TSrc^A \mid A \in N \setminus S, TSrc^A \leftarrow \emptyset\}$ 
4:   for all  $v \in Src$  do
5:      $TSrc_{v,v}^S \leftarrow true$ 
6:   for all  $A \rightarrow x \in P$  do
7:     for all  $(v, x, to) \in E$  do
8:        $T_{v,to}^A \leftarrow true$ 
9:   while  $T$  or  $TSrc$  is changing do
10:    for all  $A \rightarrow BC \in P$  do
11:       $M \leftarrow TSrc^A * T^B$ 
12:       $T^A \leftarrow T^A + M * T^C$ 
13:       $TSrc^B \leftarrow TSrc^B + TSrc^A$ 
14:       $TSrc^C \leftarrow TSrc^C + GETDST(M)$ 
15:
16: function GETDST( $M$ )
17:    $A \leftarrow \emptyset$ 
18:   for all  $(v, to) \in V^2 \mid M_{v,to} = true$  do
19:      $A_{to,to} \leftarrow true$ 
20:   return  $A$ 

```

Listing 2 Context-free path querying algorithm

```

1: function MULTISrcCFPQSMART( $index = (D, G, T, TSrc), Src$ )
2:    $TNewSrc^A \leftarrow \{TNewSrc^A \mid A \in N \setminus S, TNewSrc^A \leftarrow \emptyset\}$ 
3:   for all  $v \in Src \mid index.TSrc_{v,v} = false$  do
4:      $TNewSrc_{v,v}^S \leftarrow true$ 
5:   while  $index.T$  or  $TNewSrc$  is changing do
6:     for all  $A \rightarrow BC \in P$  do
7:        $M \leftarrow TNewSrc^A * index.T^B$ 
8:        $index.T^A \leftarrow index.T^A + M * index.T^C$ 
9:        $TNewSrc^B \leftarrow TNewSrc^B + TNewSrc^A \setminus index.TSrc^B$ 
10:       $TNewSrc^C \leftarrow TNewSrc^C + GETDST(M) \setminus index.TSrc^C$ 

```

3.1 Implementation Details

Algo implementation details: python, graphBLAS, SuiteSparse, smthng else? Two versions.

Link to the repo (should be migrated to JB Research first)

3.2 Algorithm Evaluation

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

Evaluation setup. Hardware basic description.

Graphs and queries from CFPQ_Data³ 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 necessary to choose an appropriate graph database. It was shown by Arseniy Terekhov et al. in [?] that matrix-based algorithm can be naturally 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 extension 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 extension⁴, proposed by Tobias Linddaaker, but this syntax does not implement yet in Cypher parsers.

RedisGraph database supports subset of Cypher language and uses `libcypher-parser`⁵ library to parse queries. We extend this library by introducing new syntax proposed !!! We implement⁶ full extension, not only part which is necessary for simple CFPQ.

Main feature which allows one to specify context-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 example !!!

Examples of queries. Description of examples.

4.2 RedisGraph Extending

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

Limits, restrictions, examples, etc.

4.3 Evaluation

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

Regular queries. 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 queries. We demonstrate, that our solution allows one evaluate not only context-free queries, 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 queries is possible in practice by using CFPQ algorithm, as far as regular queries is a partial case 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 problem is to use tensor-based algorithm [?].

Another important task is to compare non-linear-algebra-based approaches to multiple-source CFPQ with the proposed solution. In [?] Jochem Kuypers 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 algorithm 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 required subgraph.

³!!!

⁴Formal 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.

⁵The `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.

⁶The modified `libcypher-parser` library with support of syntax for path patterns: <https://github.com/YaccConstructor/libcypher-parser>. Access date: 19.07.2020.