Evaluation of Matrix-Based Context-Free Path Quering Algorithm

Iaroslav Sokolov
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
sokolov.yas@gmail.com

Nikita Mishin
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
mishinnikitam@gmail.com

Egor Spirin
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
egor@spirin.tech

Vladimir Kutuev
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
vladimir.kutuev@gmail.com

Egor Nemchinov
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
nemchegor@gmail.com

Sergey Gorbatyuk
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
sergeygorbatyuk171@gmail.com

Semyon Grigorev
semen.grigorev@jetbrains.com
Saint Petersburg State University
7/9 Universitetskaya nab.
St. Petersburg, Russia 199034
JetBrains Research
Universitetskaya emb., 7-9-11/5A
St. Petersburg, Russia 199034

ABSTRACT

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference format:
Iaroslav Sokolov, Nikita Mishin, Egor Spirin, Vladimir Kutuev, Egor Nemchinov, Sergey Gorbatyuk, and Semyon Grigorev. 2018. Evaluation of Matrix-Based Context-Free Path Quering Algorithm. In *Proceedings of Woodstock '18: ACM Symposium on Neural Gaze Detection, Woodstock, NY, June 03–05, 2018 (Woodstock '18),* 3 pages.
https://doi.org/10.1145/1122445.1122456

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Graph querying, Context-Free Path Querying (CFPQ), applications in different areas. Performance is important for practical tasks.

Matrix-based algorithm. Pretty simple. Performance problems. CPU/GPGPU based implementation. Investigate and compare.

There is no publically available standartized dataset for algorithms evaluation. We collect some data and propose possible candidate for it.

Research question: comparison of differend implementations of matrix-based CFPQ. We implement and compare peformance.

Contribution

- Implementation. Source code is available on GitHub:!!!!!!
- Evaluation
- Dataset for evaluation. Available. Data format. Reference values.

This paper is organized as follows. !!!!

2 MATRIX-BASED ALGORITHM FOR CFPQ

Matrix-based algoritm for CFPQ was proposed by Rustam Azimov [2]. This algorithm can be expressed in few lines of code in terms of matrices operations, and it is a sufficient advantage for implementation. It was shown that GPGPU utilization for queryes evaluation can significantly improve performance in comparison with other implementations [2] even float matrices used instead of boolean matrices.

Pseudocode of the algorithm is presented in listing 1.

Algorithm 1 Context-free path quering algorithm

function CONTEXTFREEPATHQUERYING(D, G)
 $n \leftarrow$ the number of nodes in D
 $E \leftarrow$ the directed edge-relation from D
 $P \leftarrow$ the set of production rules in G
 $T \leftarrow$ the matrix $n \times n$ in which each element is $\varnothing$
 for all $(i, x, j) \in E$ **do** ▷ Matrix initialization
  $T_{i,j} \leftarrow T_{i,j} \cup \{A \mid (A \rightarrow x) \in P\}$
 end for
 while matrix T is changing **do**
  $T \leftarrow T \cup (T \times T)$ ▷ Transitive closure calculation
 end while
 return T
end function

Here $D = (V, E)$ be the input graph and $G = (N, \Sigma, P)$ be the input grammar. Each cell of the matrix T contains the set of nonterminals such that $N_k \in T[i, j] \iff \exists p = v_i \dots v_j$—path in D, such that $N_k \overset{*}{\underset{G}{\Rightarrow}} \omega(p)$, where $\omega(p)$ is a word formed by labels along path p. Thus, this algorithm solves reachability problem, or, according Hellings [3], process CFPQs by using relational query semantics.

As you can see, performance-critical part of this algorithm is a matrix multiplication. Note, that the set of nonterminals is finite, we can represent the matrix T as e set of boolean matrices: one for each nonterminal. In this case the matrix updeta operation be $T_{N_i} \leftarrow T_{N_i} + (T_{N_j} \times T_{N_k})$ for each production $N_i \rightarrow N_j\ N_k$ in P. Thus we can reduse CFPQ to boolean matrices multiplication. After such transfromation we can apply the next optimization: we can skip update if there are no changes in the matrices T_{N_j} and T_{N_k} at the previous iteration.

Thus, the most important part is efficient implementation of operations over boolean matrices, and in this work we compare effects of utilization of different approaches to matrices multiplication. All our implemetations are based on the optimized version of the algorithm.

3 IMPLEMENTATION

We implement matrix-based algorithm fo CFPQ by using a number of different programming languages and tools. Our goal is to investigate effects of the next features of implmentation.

- **GPGPU utilization.** It is mell-known thet GPGPUs are sutable for matrices operations, but performance of whole solution depends on task details: overhead on data transferring may negate effect of parallel computations. Moreover, it is believed that GPGPUs is not sutable boolean calculations [**?**]. Can GPGPUs utilization for CFPQ improve performance in comparison with CPU version?
- **Existing libraries utilization** is a good practice in software engeneering. Is it possible to achaive highe performance by using existing libraries for matrices operations or we need to create own solution to get more control?
- **Low-level programming**. GPGPU programming is traditionally low-level programming by using C-based languages (CUDA C, OpenCL C). On the other hand, there are number of approaches to create GPGPU-based solution by ysing such high-level languages as a Python. Can we get high-performance solution by using such approaches?
- **Sparce** Real graphs often are sparse.

We provide next implementations to compare different approaces to algorithms implementation.

- CPU-based solutions
  - Python + Scipy [4] (sparse matrices)
  - C + m4ri [1] (4 russian method)
- GPGPU-based solutions
  - CUDA C, manual implemenattion of 4 russian metod.
  - CUDA C, manual implementation of naive boolean matrix multiplication
  - Pyton + numba[1] manual implementation of naive boolean matrix multiplication

Brief overview of approaches.

Generic notes on optimizations. Notes on data transferring. On matrix changes tracking (we should multiply pair of matrices only if one of them changed in last iteration)

3.1 m4ri

Descriprion of impl 1 [1]

3.2 Pyton sparse CPU

Descriprion of impl 2

3.3 CUDA naive

Descriprion of impl 3

3.4 CUDA 4 russian method

Descriprion of impl 4

3.5 Python + CUDA

Descriprion of impl 5

3.6 Smth else?

Descriprion of impl n

4 DATASET DESCRIPTION

The datatset includes the next data.

- RDFs [**?**]
- Worst case [**?**]
- Cycle to full
- Sparse graphs [**?**]

All data is presented in text-based format to simplify usage in different environments. Grammars are stored in the files with `yrd` extension. Graphs are stored in the files with `txt` extension.

Data organized as follows. Folders and subfolders

5 EVALUATION

We evaluate all described implementattions on all data and queries from presented dataset.

For evaluation we use PC with the next characteristics.

[1]Numba is a JIT compiler which supports GPGPU for subset of Python programming. Offical page: http://numba.pydata.org/. Access date: 03.05.2019

$$S \to SS$$

Figure 1: grammar 1

$$S \to SS$$

Figure 2: grammar 2

$$S \to SS$$

Figure 3: grammar 1

$$S \to SS$$

Figure 4: grammar 1

- OS
- CPU
- RAM
- GPU
- Libs versions
- Pyton runtime

Compiler options, Python runtime, etc.

RDFs and two tipes of queries. Form paper []

Worst case and same generation query.

Cycle which fills to full graph and two queryes.

Results: tables, graphics, etc

6 DISCUSSION

Discussion of evaluation results.

7 CONCLUSION AND FUTURE WORK

We present !!!

Our evaluation shows that !!!

First direction for future research is a more detailed CFPQ algorithms investigation. We should do More evaluation on sparse matrices on GPGPUs.

Also it is nesessary to implement and evaluate solutions for graphs which is not fit in RAM. There is a set of technics for huge matrices multiplication. Is it possible to dopt it for CFPQ

Another direcion is a dataset improvement. More data. More grammars/queries.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

[1] Martin Albrecht and Gregory Bard. 2019. *The M4RI Library*. The M4RI Team. https://bitbucket.org/malb/m4ri

[2] 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

[3] Jelle Hellings. 2014. Conjunctive context-free path queries. In *Proceedings of ICDT'14*. 119–130.

[4] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. http://www.scipy.org/ [Online; accessed <today>].