Context-Free Path Querying via Matrix Equations

Yuliya Susanina
Saint Petersburg State University
St. Petersburg, Russia
JetBrains Research
St. Petersburg, Russia
st049970@student.spbu.ru

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

ABSTRACT

Graph data model is very popular nowadays (bioinformatics, static code analysis). Context-free path querying (CFPQ) underlies many problems related to this data model. The importance of developing highly efficient approaches for solving CFPQ problem connected with a large amount of data to process. Computational mathematics may be very useful due to the meaningful theoretical basis and multiple constantly improving implementations. We show how to reduce GFPQ evaluation to solving the systems of matrix equations over $\mathbb{R}$ and also demonstrate the applicability of our approach to the real-world data.

CCS CONCEPTS

• **Information systems → Query languages**; • **Theory of computation → Formal languages and automata theory**.

KEYWORDS

context-free path querying, graph databases, context-free grammar, nonlinear matrix equations, newton method

ACM Reference Format:
Yuliya Susanina and Semyon Grigorev. 2018. Context-Free Path Querying via Matrix Equations. In *Proceedings of Woodstock '18: ACM Symposium on Neural Gaze Detection (Woodstock '18)*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Context-free path querying (CFPQ) is becoming more popular in many areas, e. g. bioinformatics [1], graph databases [5] or static code analysis [8]. A query is presented as a context-free grammar and all possible paths (strings) are set as a directed labeled graph. This type of query increases the expressive power of commonly used regular expressions and therefore forms a promising research area. The computation of such queries is parsing all paths in the input graph with respect to the given grammar. The result of the CFPQ evaluation is a set of triples (A, m, n), such that a path from the node m to the node n exists in the given graph and a string obtained by concatenating the labels of the edges along this path is derived from nonterminal A of the given context-free grammar.

A large amount of data is typical for most CFPQ application areas, so an especially pronounced problem is the search and development of high-performance algorithms. The matrix-based algorithms, for example, [3], are the most potential for practical tasks. The possibility of applying various computing techniques to speed up matrix calculations, such as GPGPU or sparse matrix representation allows to improve the performance on the real-world data [6]. But these algorithms, just like the others, suffer from computational problem issues and can be low-speed in some cases.

There are also several reasons to consider the applicability of numerical linear algebra and computational mathematics for CFPQ. The benefits of matrix-based CFPQ algorithms (e.g. the utilization of parallel techniques) will remain. Well-known linear algebra operations, such as matrix inversion or decomposition, can be used in the creation of better algorithms. For instance, there are several successful results of applying linear algebra methods to logic programming [2, 7]. Also, approximate computational methods can accelerate CFPQs processing. And most importantly, the popularity of artificial intelligence techniques pushed the development and improvement of many efficient libraries for numerical computing.

In this article, we modify the matrix-based algorithm mentioned above and reduce GFPQ to solving the systems of Boolean matrix equations. And then we propose a new approach for CFPQs processing, based on solving the systems of equations in the set of real numbers $\mathbb{R}$. We also assess the feasibility of using both accurate and approximate methods of computational mathematics. The evaluation of our approach on a set of conventional benchmarks shows its applicability.

2 BACKGROUND

Context-free grammar (CFG) is a triple $G = (N, \Sigma, R)$, where N is a set of nonterminal symbols, Σ is a set of terminal symbols and R is a set of productions of the followings form: $A \Rightarrow \alpha$, $\alpha \in (N \cup \Sigma)^*$. $\mathcal{L}(G_S)$ denotes a language specified by CFG G with respect to $S \in N$: $\mathcal{L}(G_S) = \{\omega \mid S \Rightarrow_G^* \omega\}$.

Directed graph is a triple $D = (V, E, \sigma)$, where V is a set of vertices, $\sigma \subseteq \Sigma$ is a set of labels, and a set of edges $E \subseteq V \times \sigma \times V$. Denote a path from the node m to the node n in graph D as $m\lambda n$, where λ is the unique word, obtained by concatenating the labels of the edges along this path. P is a set of all paths in D.

For a graph D and a CFG G, we define *context-free relations* $R_A \subseteq V \times V$ for each $A \in N$: $R_A = \{(m, n) \mid m\lambda n \in P, \lambda \in \mathcal{L}(G_A)\}$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Woodstock '18, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

Matrix-based algorithm, proposed by Rustam Azimov [3], processes CFPQs by using relational query semantics [4] and offloads the most time-consuming operations onto Boolean matrix multiplication, thus allowing significantly increase the performance [6], but it takes time $O(|V|^{4+\omega})$.

3 EQUATION-BASED APPROACH

In this section we transform matrix-based algorithm. We replace the calculation of matrices' products by the computation of several matrix equations.

For each $X \in (N \cup \Sigma)$ we create a Boolean matrix T_X. All matrices corresponding to nonterminals are fixed according to the input graph: $((T_a)_{ij} = 1 \iff (v_i, a, v_j) \in E)$ for $a \in \Sigma$. And for each $A \in N$: $((T_A)_{ij} = 1 \iff (i,j) \in R_A)$.

Let's consider a simple CFG $G_1 : S \to aSb \mid ab$ and its representation in Chomsky normal form: $G_1 : S \to AS_1 \mid AB$; $S_1 \to SB$; $A \to a$; $B \to b$. In the matrix-based algorithm we construct a matrix transitive closure in a while loop and on each iteration Boolean matrices is changing in 3 calculations (rules $A \to a$ and $B \to b$ are not used after matrices initialization). It can be replaced into a simple iterative process:

$$\left.\begin{array}{l} T_{S_1} = T_{S_1} + T_S T_b \\ T_S = T_S + T_a T_{S_1} \\ T_S = T_S + T_a T_b \end{array}\right\} \Rightarrow \begin{array}{l} T_S^0 = 0 \\ T_S^{k+1} = T_a T_S^k T_b + T_a T_b \end{array}$$

$\{T_S^k\}$ is a monotonically increasing series of Boolean matrices. It converges and the limit T_S^* is the least solution of Boolean matrix equation:

$$T_S = T_A T_S T_B + T_A T_B$$

Unfortunately, this modification does not entail any meaningful improvements. But we can consider another equation over $\mathbb{R}$:

$$\mathcal{T}_S = \epsilon(T_A \mathcal{T}_S T_B + T_A T_B)$$

And the corresponding matrix series $\{\mathcal{T}_S^k\}$, which converges to $\mathcal{T}_S^*$ when $\mathcal{T}_S^k \leq \mathbf{1}$ as a a monotonically increasing series of matrices with an upper bound:

$$\mathcal{T}_S^0 = 0$$
$$\mathcal{T}_S^{k+1} = \epsilon(T_A \mathcal{T}_S^k T_B + T_A T_B).$$

It can be proved that $((\mathcal{T}_S^{k+1})_{ij} > 0 \iff (T_S^{k+1})_{ij} = 1)$ and $ceil(\mathcal{T}_S^*) = T_S^*$, where $ceil(x)$ returns the smallest integer not less than x.

So, each rule of the following form $X \to V \ldots W \mid \cdots \mid Y \ldots Z$, where $X \in N$, $V, W, \ldots, Y, Z \in (N \cup \Sigma)$ can be replaced by equation: $T_X = \epsilon_X(T_V \cdot \ldots \cdot T_W + \cdots + T_Y \cdot \ldots \cdot T_Z)$, where ϵ_X is chosen such that $\mathcal{T}_X^k \leq \mathbf{1}$ for each k.

Mostly we deal with the systems of matrix equations. Let's briefly consider the possible cases of these systems.

Linear Equations

If the input CFG is linear, each equation of the system is of the form of generalized Sylvester equation: $\sum_{i=1}^{k} A_i X B_i = C$. For $k = 1, 2$ we can solve it in $O(|V|^3)$. Otherwise, for $k > 2$ it can be reduced using Kronecker product to solving a linear system $Ax = b$, where A is a matrix of size $(|V|^2 \times |V|^2)$ and time required to compute its solution is $O(|V|^6)$ or $O(|V|^{4\omega+2})$ with more efficient matrix multiplication algorithms. The use of sparse matrix representation can be very efficient for solving the equations of this type.

For the system of equations, we construct the dependency graph D_G for nonterminals of the given grammar G and split the set of the equations into the disjoint subsets accordingly to the set of strongly connected components in D_G, which can be found in $O(|V| + |E|)$. Then we solve our system in stages, for each subset.

Nonlinear Equations

For the nonlinear case we can rewrite each equation of the form $X = \Psi(X)$ to the equivalent $F(X) = X - \Psi(X) = 0$ and use Newton's method for nonlinear functions root finding:

$$F(X) = 0, X_0$$

$$X_{i+1} = X_i - (F'(X_i))^{-1} F(X_i) \iff \begin{cases} F'(X_i) H_i = -F(X_i) \\ X_{i+1} = X_i + H_i \end{cases}$$

Here X_0 is an initial guess, in our case $X_0 = \mathbf{0}$, as our solution is a matrix consists of small positive numbers. The convergence of this method can be quadratic which allows finding the solution significantly faster. Even as it is necessary to solve an equation for H_i (also generalized Sylvester equation) on each iteration step, the majority of high-performance implementations do not compute the Jacobian inverse and use its approximate value.

The main difficulty in using Newton's method is choosing an appropriate ϵ, to ensure the least positive solution for nonlinear equations ϵ must be smaller than $\frac{1}{|V|}$.

4 EVALUATION

The equation-based approach for CFPQ was implemented. We evaluated **Query 2** [3]. The equation constructed for this query were resolved on the CPU by two different ways using Python package *scipy*: **sSLV** — solving as a sparse linear system using *spsolve* and **dNWT** — funding roots of a function using *optimize.newton_krylov*.

Table 1: Evaluation results for Query 2 (in ms)

Ontology	|V|	dGPU	sCPU	dNWT	sSLV	sGPU
bio-meas	341	276	91	284	35	24
people-pets	337	144	38	73	49	6
funding	778	1246	344	502	184	27
wine	733	722	179	791	171	6
pizza	671	943	256	334	161	23

We compare (Table 1) the results with the first matrix-based algorithm implementations described in [3]. Our approach demonstrates that it can be applied on real-world data as well as the matrix-based algorithm. Moreover, we can improve the performance by the utilization of parallel techniques for matrix operations.

5 CONCLUSION AND FUTURE WORK

We proposed a new approach for CFPQs processing, based on solving the systems of equations over $\mathbb{R}$. The evaluation of our approach on a set of conventional benchmarks showed its practical applicability on real-world data.

The directions for future research are high-performance implementation using GPGPU or other parallel techniques. Also, it seems interesting to determine the subclasses of (system of) polynomial equations whose solution can be reduced to CFPQ and try to construct bidirectional reduction between CFPQ and these subclasses, thereby finding efficient solutions for both these problems at a time.

ACKNOWLEDGMENTS

The research was supported by the Russian Science Foundation grant 18-11-00100 and a grant from JetBrains Research.

REFERENCES

[1] James WJ Anderson, Ádám Novák, Zsuzsanna Sükösd, Michael Golden, Preeti Arunapuram, Ingolfur Edvardsson, and Jotun Hein. 2013. Quantifying variances in comparative RNA secondary structure prediction. *BMC bioinformatics* 14, 1 (2013), 149.

[2] Yaniv Aspis. 2018. *A Linear Algebraic Approach to Logic Programming*. Ph.D. Dissertation. Imperial College London.

[3] 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)*. ACM, 5.

[4] Jelle Hellings. 2015. Querying for paths in graphs using context-free path queries. *arXiv preprint arXiv:1502.02242* (2015).

[5] A. Mendelzon and P. Wood. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Computing* 24, 6 (1995), 1235–1258.

[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] Taisuke Sato. 2017. A linear algebraic approach to Datalog evaluation. *Theory and Practice of Logic Programming* 17, 3 (2017), 244–265.

[8] Qirun Zhang, Michael R Lyu, Hao Yuan, and Zhendong Su. 2013. Fast algorithms for Dyck-CFL-reachability with applications to alias analysis. In *ACM SIGPLAN Notices*, Vol. 48. ACM, 435–446.