Stochastic Context-Free Path Querying by Matrix Multiplication

Yuliya Susanina
Saint Petersburg State University
JetBrains Research
St. Petersburg, Russia
jsusanina@gmail.com

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

ABSTRACT

Stochastic context-free path querying (SCFPQ) is a way to find the probability of paths with some context-free constrains in a given graph. In this article we formulate two questions of SCFPQ: most probable paths problem and all-paths probability problem. We also adapt main context-free path querying algorithms based on matrix operations and linear algebra to solve these problems.

CCS CONCEPTS

• **Information systems** → **Query languages for non-relational engines**; • **Mathematics of computing** → *Computations on matrices*; • **Theory of computation** → *Formal languages and automata theory*.

KEYWORDS

context-free path querying; graph databases; context-free grammar; stochastic context-free grammars; nonlinear matrix equations; Newton's method

ACM Reference Format:
Yuliya Susanina and Semyon Grigorev. 2020. Stochastic Context-Free Path Querying by Matrix Multiplication. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3318464.3384400

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06.
https://doi.org/10.1145/3318464.3384400

1 INTRODUCTION

Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important.

Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important.

Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important. Stochastic context-free path querying is very important.

In this paper we define the stochastic context-free path querying problems and also adapt main context-free path querying algorithms based on matrix operations and linear algebra to solve these problems.

2 BACKGROUND

2.1 Context-Free Path Querying

Context-free grammar (CFG) is a triple $G = (N, \Sigma, R)$, where N is a set of nonterminals, Σ is a set of terminals and R is a set of productions of the followings form: $A \to \alpha, \alpha \in (N \cup \Sigma)^*$. Context-free grammar $G_S = (\Sigma, N, R, S)$ (with the selected start nonterminal S) is said to be in *Chomsky normal form* if all productions in R are of the form: $A \to BC$, $A \to a$, or $S \to \varepsilon$, where $A, B, C \in N, a \in \Sigma, \varepsilon$ is an empty string. $\mathcal{L}(G_S)$ denotes a language specified by CFG G with respect to $S \in N$: $\mathcal{L}(G_S) = \{\omega \mid S \Rightarrow^*_G \omega\}$.

Labeled directed graph (LDG) 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.

Context-free path querying (CFPQ) problem with relational semantics (according Hellings [2]) is for a LDG D and a CFG G to find *context-free relations* $R_A \subseteq V \times V$ such that $\forall A \in N$: $R_A = \{(m, n) \mid m\lambda n \in P, \lambda \in \mathcal{L}(G_A)\}$. Every relation R_A is finite, so it can be represented as a Boolean matrix: $(T_A)_{i,j} = 1 \Leftrightarrow (i, j) \in R_A$.

2.2 CFPQ via Matrix Multiplication

Matrix-based algorithm, proposed by Rustam Azimov [1], processes CFPQs by using relational query semantics [3]. It constructs a parsing table T of size $|V| \times |V|$ for an input graph $D = (V, E)$ and grammar $G = (N, \Sigma, R)$ in Chomsky normal form. Each element of T contains the set of nonterminals such that $A \in T_{i,j} \iff \exists p \in R_A$.

For each vertices i and j we initialize $T_{i,j} = \{A \mid (i, a, j) \in E, A \to a \in R\}$. Then, the computations of parsing table T happens through the calculation of matrix transitive closure: $M^* = M^{(1)} \cup M^{(2)} \ldots$, where $M^{(1)} = M$, $M^{(k)} = M^{(k-1)} \cup (M^{(k-1)} \times M^{(k-1)})$ for $k > 1$. Also we can represent parsing table T as a set of Boolean matrices of size $|V| \times |V|$ for each $A \in N$. So, we can replace the computation of transitive closure $T = T \cup (T \times T)$ to several Boolean matrix multiplications $T_A = T_A + T_B T_C$ for each $A \to BC \in R$.

This algorithm can be effectively applied to real-world data with implementation based on parallel techniques and high-performance libraries [4], but it takes time $O(|N|^3 |V|^2 (BMM(|V|) + BMU(|V|)))$.

2.3 CFPQ via Systems of Matrix Equations

Equation-based approach was proposed in [7]. Despite of the fact that the main result of this work was a reduction of context-free path querying to systems of nonlinear matrix equation solving, there is also a reduction to solving Boolean matrix equations. Almost like in the matrix-based approach we create a Boolean matrix T_E for each terminal and nonterminal $E \in \Sigma \cup N$. After that for each production of form

$$N_i \to \beta^0_0 \ldots \beta^0_k \mid \ldots \mid \beta^l_0 \ldots \beta^l_m, \beta^i_j \in \Sigma \cup N$$

we can create an equation

$$T_{N_i} = T_{\beta^0_0} \cdot \ldots \cdot T_{\beta^0_k} + \ldots + T_{\beta^l_0} \cdot \ldots \cdot T_{\beta^l_m}.$$

The obtained system of matrix equations can be solved by a naïve iterative process. This approach can be considered as (Chomsky normal form)-free version of matrix-based algorithm and has the same time complexity.

2.4 Stochastic Context-Free Grammars

Stochastic context-free grammar (SCFG) G_Θ is a pair (G, Θ), where G is a context-free grammar (N, Σ, R) and $\Theta : R \to (0, 1]$, such that $\forall A \in N : \sum_{i=1}^{n_A} \Theta(A \to \alpha_i) = 1$, n_A is a number of productions with nonterminal A in their left side. Define a set of all (left-most) derivations or parse trees under the grammar G_Θ as $\mathcal{T}_{G_\Theta}$ and all parse trees of string s as $\mathcal{T}_{G_\Theta}(s)$.

The probability of the derivation τ_s of the string s is the product of the probability application function of all the rules used in the derivation τ_s: $Pr_{G_\Theta}(s, \tau_s) = \prod_{\forall p \in \tau_s} \Theta(p)$. The probability of the string s (if there are more than one parse tree) is $Pr_{G_\Theta}(s) = \sum_{\forall \tau_s} Pr_{G_\Theta}(s, \tau_s)$.

Two main problems in the theory of stochastic context-free grammars are to find a parse tree with the maximum probability for the given string — $\max_{\tau_s} Pr_{G_\Theta}(s, \tau_s)$ and to compute the probability of the string — $Pr_{G_\Theta}(s)$. In this article we rephrase these two problems to own case, when the input data is a graph instead of a string.

3 STOCHASTIC CONTEXT-FREE PATH QUERYING

In this section we make an effect to formulate the problem of stochastic context-free path querying and its main questions.

3.1 Rule-Weighted Grammar

For a given SCFG G_Θ and a LDG D define the maximum probability of the path $m\lambda n$ as the maximum probability of the string λ along this path — $\max_{\tau_\lambda} Pr_{G_\Theta}(\lambda, \tau_\lambda)$ and the probability of the path $m\lambda n$ as the probability of the string λ along this path — $Pr_{G_\Theta}(\lambda)$. Note that if $\lambda \notin \mathcal{L}(G_\Theta)$, than $Pr_{G_\Theta}(\lambda)$ is equal to 0. The maximum probability between two nodes m and n (the maximum probability to reach node n from node m) is the maximum probability among all paths between these two nodes — $\max_{m\lambda n \in D}(\max_{\tau_\lambda} Pr_{G_\Theta}(\lambda, \tau_\lambda))$. The probability between two nodes m and n (the probability to reach node n from node m) is the sum of the probabilities of all paths $m\lambda n$ in D — $Pr_{G_\Theta}(m, n) = \sum_{m\lambda n \in D} Pr_{G_\Theta}(\lambda)$.

Most probable paths problem is for a given SCFG G_Θ and a LDG D, to find the maximum probabilities between all pairs of nodes m and n.

All-paths probability problem for a given SCFG G_Θ and a LDG D, to find the probabilities between all pairs of nodes m and n.

We have considered how to adapt the two main problems of stochastic parsing to the problem of stochastic CFPQ. We are interested in finding the most likely paths in the graph or in the probability of starting at the node m and choosing a random direction of movement in the graph to eventually reach the node n.

3.2 Edge-Weighted Graph (undeveloped)

In this case, we consider ordinary context-free grammar, but the weights are on the edges of the given labeled graph to influence which paths in the graph we find more preferable. The focus is shifting from the paths which contains some desirable patterns given by stochastic context-free grammar to finding paths with the same patterns, but giving preference to some specific of these paths.

Let w is a weight function of D, which assign each edge e of D a weight $w(e)$, also w is a stochastic weight function if $\sum_{e=(m,\_,\_)} w(e) = 1$. Hereinafter, by weight, we mean the stochastic weight. For a given CFG G and a weighted LDG (WLDG) D_w define the probability of the path $m\lambda n$ is the product of all the edges which belongs to this path: $Pr_G(m\lambda n) = \prod_{\forall e \in m\lambda n} w(e)$. The maximum probability between two nodes m and n is the maximum probability among all paths between these two nodes — $\max_{m\lambda n \in D}(Pr_G(m\lambda n))$. Similarly, the probability between two nodes m and n is the sum of the probabilities of all paths $m\lambda n$ in $D - Pr_G(m, n) = \sum_{m\lambda n \in D} Pr_G(m\lambda n))$.

So, we can formulate the same two problems.

Most probable paths problem is for a given CFG G and a WLDG D_w, to find the maximum probabilities between all pairs of nodes m and n.

All-paths probability problem for a given CFG G and a WLDG D_w, to find the probabilities between all pairs of nodes m and n.

To sum up, in this section we consider two variants of stochastic context-free path querying problem, when the stochastic grammar is given or when the edges of the graph are weighted. It must be mentioned that there is a special case of the intersection of these two problems. In [5] Ponty shows that rule-weighted grammars and terminal-weighted grammars have the same expressivity when the grammars are unambiguous. So, if the edges with the equal labels have the same weight and the grammar is unambiguous, we can reduce the problem of SCFPG with a rule-weighted grammar to SCFPQ with an edge-weighted graph and vice versa.

Algorithm 1 General matrix-based algorithm for SCFPG

```
1: function contextFreePathQuerying(D, G)
2:     n ← a number of nodes in D
3:     T ← a set of matrices T_A ∀A ∈ N of size n × n
4:           full of zeroes
5:     for all (i, x, j) ∈ E do
6:         for all (A → x) ∈ R do
7:             T_A[i, j] ← Θ(A → x)
8:     while matrices T is changing do
9:         matrices T computation
```

4 MATRIX-BASED ALGORITHM

In this section we modify a matrix-based algorithm for two defined above problems of stochastic context-free path querying: most probable paths problem and all-paths probability problem. Algorithm 1 shows a general form of a matrix-based approach for stochastic context-free path querying. In the case of the matrix-based algorithm the stochastic context-free grammar is in Chomsky normal form. For each problem we define a specific function in line 8. Other lines is common for both problems.

4.1 Most Probable Paths

The function for most probable paths problem is:

$$T_A[i,j] = (f(T_B, T_C))[i,j] = \tag{1}$$
$$= \Theta(A \to BC) \max_{0 \le k < n} T_B[i,k] T_C[k,j]$$
or
$$= \max_{\substack{A \to BC \\ 0 \le k < n}} \Theta(A \to BC) T_B[i,k] T_C[k,j]$$

In this case the result of our algorithm is the probability parsing matrices T_A for each nonterminal A. The elements of this matrices $T_A[i, j]$ is equal to the maximum probability between nodes i and j, the maximum probability of some path $i\lambda j$, where $\lambda \in \mathcal{L}(G_\Theta)$.

Theorem 4.1. *For each nonterminal A of the matrix-based algorithm for the most probable paths problem $T_A[i, j]$ contains the maximum probability between two nodes i and j.*

Proof. (By induction)

Base case. Show that the elements of matrix $T_A[i, j]$ $\forall A \in N$ contains the maximum probability between paths, which derivation tree of the word along this path has a height equal to 1. Since the derivation of height 1 can be obtained by the rule of the form $A \to x$ and for each nonterminal A and terminal x has the unique rule of such form, then there is a

unique (maximum) weight, which is correctly filled in lines 5-7.

Inductive step. Assume the elements of matrix $T_A[i, j]$ $\forall\ A \in N$ after the $k-1^{th}$ iteration contain the most probable path, which derivation tree of the word along this path has a height less than k. Show that during the k^{th} iteration we consider the maximum probability paths with derivation trees of height less than $k+1$.

The path from node i to node j with the derivation from nonterminal A of height k can be obtained by applying the rule of the form $A \rightarrow BC$, where exist two paths — from node i to node l derived from nonterminal B and from node $l+1$ to node j derived from nonterminal C with the derivation height not exceeding $k-1$ (for some $0 \leq l < n$) .

So, by the inductive hypothesis $T_B[i, l]$ and $T_C[l+1, j]$ contains the maximum probabilities of these required paths. In formula 1 we consider all such pairs of paths (for all possible l) and find the maximum probable concatenation of them. And as the defined probability is a product of the probabilities of all the rules, then we must multiply our chosen maximum by the probability of the applying rule.

□

Theorem 4.2. *The time complexity of the proposed algorithm is* $O(|N|^3|V|^5)$.

Proof. As the probability of the word of length m cannot exceed the probability of the word of length n, where $m < n$ (!!!or about cycles!!!), the maximum number the matrix T_A $\forall A \in N$ can be changed $|V|^2$ times and consequently the maximum number of iterations is $|N||V|^2$. In formula 1 to compute $T_A[i, j]$ $|N||V|$ operations is needed. So, the total time complexity is $O(|N|^3|V|^5)$. □

4.2 All-Paths Probability

The following function corresponds to the case of all-paths probability problem:

$$T_A[i,j] = (f(T_B, T_C))[i,j] = \tag{2}$$
$$= T_A[i,j] + \Theta(A \rightarrow BC) \sum_{k=0}^{n-1} (T_B[i,k] T_C[k,j])$$
$$= T_A[i,j] + \Theta(A \rightarrow BC)\ (T_B T_C)[i,j]$$

$$T_A = T_A + \Theta(A \rightarrow BC)\ T_B T_C \tag{3}$$

Here, the probability parsing matrices T_A for $A \in N$ contains the general probability between two nodes, that is the sum of all paths probabilities.

It is worth mentioning that if the input graph has cycles, than the number of paths is infinite, so it is not possible to find the accurate probability results. So, the only way to obtain preferable result is to apply the approximate methods of computational mathematics.

Theorem 4.3. *For each nonterminal A of the matrix-based algorithm for the all-paths probability problem converges and* $T_A[i, j]$ *contains the probability between two nodes i and j.*

Proof. (Sketch)

Convergence. Define $T^{(k)}$ is a set of matrices T_A $\forall A \in N$ on the k^{th} iteration in lines 8-9 of the algorithm 1. Moreover, $\{T^{(k)}\}$ is a set of series of a monotonically increasing matrices for each nonterminal. A series of a monotonically increasing matrices is converges when it has an upper bound. In our case, we can note that all elements of these matrices is less that or equal to 1, as the sum of the probability of all words in the language specified by some SCFG is equal to 1 (if this SCFG is consistent), so, $T_A^{(k)}$ is bounded by a matrix full of ones for each k.

Correctness. At the $k-1^{th}$ iteration step $T_A[i, j]$ contains the sum of probabilities of all paths from node i to node j, which can be derived in the given grammar from $A \in N$ in a less than k steps (application of the productions). And at the k^{th} iteration step we add all probabilities of the paths, which derivation tree of the word along this path has a height equal to k. We consider all possible divisions of the path from i to j on two paths from i to l and from $l+1$ to j with the derivation height not exceeding $k-1$ for it and multiply the finding path by the probability of the applied production (formula 2). (!!!about inside and outside probabilities!?!) □

In addition, this algorithm can be applied to a special type of a weighted context-free grammar (WCFG) (when the rule weighted is an arbitrary positive number). WCFG is convergent if the sum of the weights of all the derivation trees in this grammar is finite. In that case, the algorithm from theorem 4.3 converges, as the matrices T_A is also bounded. In addition, Smith shows that convergent WCFG and SCFG are equally expressive [6]. So each convergent WCFG can be transformed to the equivalent SCFG (with the same probability distribution).

5 EQUATION-BASED APPROACH

5.1 All-Paths Probability

If grammar is in Chomsky Normal Form, then the function from line 9 of algorithm 1:

$$T_A[i,j] = (f(T_A))[i,j] = \tag{4}$$
$$= \sum_{A \rightarrow BC} \Theta(A \rightarrow BC) \sum_{k=0}^{n-1} (T_B[i,k] T_C[k,j])$$
$$= \sum_{A \rightarrow BC} \Theta(A \rightarrow BC)\ (T_B T_C)[i,j]$$

$$T_A = \sum_{A \to BC} \Theta(A \to BC)\ T_B T_C \tag{5}$$

If the grammar is in the arbitrary form. For each production of form

$$N_i \to \beta_0^0 \dots \beta_k^0 \mid \dots \mid \beta_0^l \dots \beta_m^l, \beta_j^i \in \Sigma \cup N$$

we can create an equation

$$T_{N_i} = \sum_{N_i \to \alpha_j} \Theta(N_i \to \alpha_j)\ T_{\beta_0^j} \cdot \dots \cdot T_{\beta_k^j}, \tag{6}$$

where $\alpha_j = \beta_0^j \dots \beta_k^j$.

Now it is necessary to talk about the convergence of the equation-based algorithm and how to solve the obtained systems of equations.

As the equation-based approach is just a modification of the matrix-based algorithm, which can handle with grammars, which are not in the Chomsky normal form, it essentially processes the same series of the matrices (mentioned in theorem 4.3's proof) on each iteration step. The algorithm from this section therefore converges and also can be applied to convergent WCFGs.

... approxomation, newton and so on.

6 CONCLUSION AND FUTURE WORK

Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work. Conclusion is a future work.

ACKNOWLEDGMENTS

The research was supported by the Russian Science Foundation grant 18-11-00100.

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] Jelle Hellings. 2014. Conjunctive Context-Free Path Queries. OpenProceedings.org. https://doi.org/10.5441/002/ICDT.2014.15

[3] Jelle Hellings. 2015. Path Results for Context-free Grammar Queries on Graphs. *CoRR* abs/1502.02242 (2015). arXiv:1502.02242 http://arxiv.org/abs/1502.02242

[4] 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

[5] Yann Ponty. 2012. Rule-weighted and terminal-weighted context-free grammars have identical expressivity. *arXiv preprint arXiv:1205.0627* (2012).

[6] Noah A Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics* 33, 4 (2007), 477–491.

[7] Yuliya Susanina. 2020. Context-Free Path Querying via Matrix Equations. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2821–2823. https://doi.org/10.1145/3318464.3384400