Context-Free Path Queries and Sparse Matrix Multiplication

Ekaterina Shemetova
The Thørväld Group
Hekla, Iceland
larst@affiliation.org

Arseniy Terekhov
Inria Paris-Rocquencourt
Rocquencourt, France

Semyon Grigorev
Rajiv Gandhi University
Doimukh, Arunachal Pradesh, India

ABSTRACT

CCS CONCEPTS

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

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Ekaterina Shemetova, Arseniy Terekhov, and Semyon Grigorev. 2018. Context-Free Path Queries and Sparse Matrix Multiplication. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

2 PRELIMINARIES

The function $nnz(A)$ denotes the number of non-zero elements in matrix A.

Let $i\pi j$ denote a unique path between nodes i and j of the graph and $l(\pi)$ denotes a unique string which is obtained from the concatenation of edge labels along the path π. For a context-free grammar $G = (\Sigma, N, P, S)$ and directed labelled graph $D = (Q, \Sigma, \delta)$, a triple (A, i, j) is *realizable* iff there is a path $i\pi j$ such that nonterminal $A \in N$ derives $l(\pi)$.

Short description of the Rustam algorithm + pseudocode **??**.

Algorithm 1 Context-free recognizer for graphs

```
1:  function contextFreePathQuerying(D, G)
2:      n ← the number of nodes in D
3:      E ← the directed edge-relation from D
4:      P ← the set of production rules in G
5:      T ← the matrix n × n in which each element is ∅
6:      for all (i, x, j) ∈ E do                ▷ Matrix initialization
7:          T_{i,j} ← T_{i,j} ∪ {A | (A → x) ∈ P}
8:      while matrix T is changing do
9:          T ← T ∪ (T × T)   ▷ Transitive closure T^{cf} calculation
10:     return T
```


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

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

3 CUBIC UPPER BOUND USING SPARSE MATRIX MULTIPLICATION

In this section we show that the Algorithm 1 has complexity $O(n^3)$ if sparse matrix multiplication is used. The reason is that one needs to compute at most $|N|n^2$ realizable triples in total during all iterations of the algorithm, whereas the number of new triples found on each i-th iteration can be relatively small. The number of operations of the i-th iteration can be reduced by multiplying the matrix $B_{(i-2)}$ containing all the previously found triples, on the matrix $A_{(i-1)}$ which has only those triples firstly obtained in the $(i-1)$-th iteraition. In other words, we have:

$$B_i = B_{i-1} + A_i, \tag{1}$$

where

$$A_i = (B_{i-2}A_{i-1} + A_{i-1}B_{i-2} + A_{i-1}A_{i-1}) - B_{i-2}. \tag{2}$$

So the following two conditions hold:

(1) $\forall i, B_{(i-2)} \cap A_{(i-1)} = \emptyset$;
(2) $\forall i, j, A_i \cap A_j = \emptyset$.

Notice that the first condition implies that one of the two multiplied matrices should be sparse, because $nnz(B_{(i-2)}) + nnz(A_{(i-1)}) \leq |N|n^2$. Also, by the second condition matrices A are pairwise disjoint, therefore $nnz(\sum_{i=1}^{n^2} A_i) \leq |N|n^2$.

The Algorithm 1 can be modified using Equations 1 and 2 instead of the naive calculation of transitive closure on every iteration. It is important that the modified agorithm has the same number of iterations in the worst case as the original one — $|N|n^2 = O(n^2)$. This is because the height of the parse tree does not exceed this value. As in the Algorithm 1, modified version derives new triples, going from leaves to the root of the parse tree.

Theorem 3.1. *The matrix B_{n^2} containing all possible realizable triples can be calculated in $O(n^3)$ time.*

Proof. The correctness of the algorithm can be easily deduced from the correctness of the Algorithm 1 [1]. Now we show the cubic time complexity of the modified algorithm. Consider the equation for calculating B_{n^2}:

$$\begin{aligned} B_{n^2} &= B_{n^2-1} + B_{i-2}A_{i-1} + A_{i-1}B_{i-2} + A_{i-1}A_{i-1} = \\ &= B_{n^2-2} + B_{i-3}A_{i-2} + A_{i-2}B_{i-3} + A_{i-2}A_{i-2} + \\ &+ B_{i-2}A_{i-1} + A_{i-1}B_{i-2} + A_{i-1}A_{i-1} = \ldots = \\ &= B_1 + B_1B_1 + \sum_{i=2}^{n^2-1} B_{i-2}A_{i-1} + \sum_{i=2}^{n^2-1} A_{i-1}B_{i-2} + \sum_{i=2}^{n^2-1} A_{i-1}A_{i-1}. \end{aligned}$$

Suppose, without loss of generality, that the matrix B_{i-2} is dense, than the matrix A_{i-1} is sparse. Using naive sparse matrix multiplication algorithm, we have that $O(nnz(A_{i-1})n)$ operations are needed for multiplication of the matrix B_{i-2} and the matrix A_{i-1}.

Let $T(AB)$ be the number of operations that need to be performed to multiply matrices A and B. Thus, the total number of operation for obtaining B_{n^2} is in

$$T(B_1B_1)+\sum_{i=2}^{n^2-1} T(B_{i-2}A_{i-1})+\sum_{i=2}^{n^2-1} T(A_{i-1}B_{i-2})+\sum_{i=2}^{n^2-1} T(A_{i-1}A_{i-1}) =$$

$$= O(n^{\omega}) + O(\sum_{i=2}^{n^2-1} nnz(A_{i-1})n) =$$

$$= O(n^{\omega}) + O(nnz(\sum_{i=2}^{n^2-1} A_{i-1})n) = O(|N|n^2n) = O(n^3).$$

□

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)* (Houston, Texas) *(GRADES-NDA '18)*. ACM, New York, NY, USA, Article 5, 10 pages. https://doi.org/10.1145/3210259.3210264