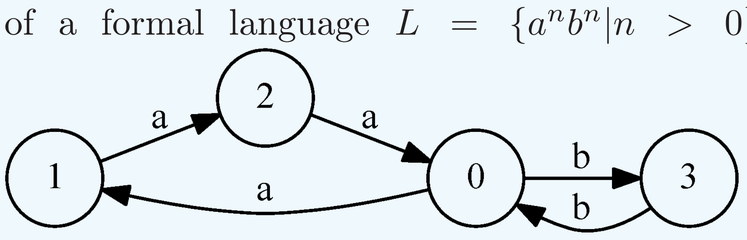
Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication

Semyon Grigorev
JetBrains Research, Saint Petersburg State University, Russia
kajigor@gmail.com

Contex-Free Path Querying

Find paths which satisfy constraints in form of a formal language $L = \{a^n b^n | n > 0\}$.



Query is a grammar for language L: $S \to a\ b \mid a\ S\ b$

The result:

$$\{(u, v) | \exists p \text{ form } u \text{ to } v : \text{word}(p) \in L\}$$

Matrix-based Algorithm [?]

T — adjacency matrix of the input graph The grammar in the normal form

$$T_{ij} = \{N \mid N \overset{*}{\Rightarrow} \omega, \omega \text{ path bw } i \text{ and } j\}$$
$$T_{ik} \times T_{kj} = \{A \mid B \in T_{ik}, C \in T_{kj}, A \to BC\}$$
$$T^{(i)} = T^{(i-1)} \cup (T^{(i-1)} \times T^{(i-1)})$$

- Can be reformulated in terms of boolean matrices multiplication
- Easy to run in parallel environments:

Results

- Dataset for CFPQ evaluation is collected and published.
  - Contains both graphs and queryes
  - Contains both real-world and syntetic graphs
- A number of CFPQ algorithms implementations are provided, evaluated and published.

Future Research

- Create open extensib;e platform for CFPQ algorithms evaluation
- Extend dataset with new data
- Implement and evaluate destributed matrix-based CFPQ algorithms
- Implement and evaluate sparse boolean matrix-based CFPQ algorithms

Implementations

Our implementations:

[Scipy] Matrix-based algorithm which uses sparse matrices from **Scipy** library (**Python**).

[M4RI] Matrix-based algorithm which uses dense matrices multiplication from **m4ri** library (Method of Four Russians, **C**)

[GPU] Matrix-based algorithm which uses our own implementation of the naïve boolean matrix multiplication in **CUDA C**

Reference implementations:

[CuSprs] Matrix-based algorithm which uses NVIDIA cuSPARSE library (**CUDA C, GPGPU**)

[CYK] CYK-based algorithm implemented in **Java** (CPU)

We need more data!

- faster
- We can handle real data
- faster
- We can handle real data
- faster
- We can handle real data

RDF			Query G_4				
Name	#V	#E	Scipy	M4RI	GPU_N	CuSprs	CYK
atm-prim	291	685	3	2	1	269	515285
biomed	341	711	3	5	1	283	420604
pizza	671	2604	6	8	1	292	3233587
wine	733	2450	7	6	1	294	4075319

Table 1: Query $s \to SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$

Scaling

Graph	Scipy	M4RI	GPU_N	CuSprs
G10k-0.001	37.286	2.395	0.215	35.937
G10k-0.1	601.182	1.050	0.114	395.393
G40k-0.001	-	97.841	8.393	-
G80k-0.001	-	1142.959	65.886	-
25000	-	33.236	5.314	-
50000	-	360.035	44.611	-
80000	-	1292.817	190.343	-

Contact us

Both dataset and implementations are available on GitHub:
`https://github.com/SokolovYaroslav/CFPQ-on-GPGPU`

Acknowledgments

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

References