LDBC TUC

Context-Free Path Querying: Obstacles on the Way to Adoption

Semyon Grigorev

JetBrains Research, Programming Languages and Tools Lab
St. Petersburg State University

16.07.2021

Who we are?

- !!!

- !!!

Formal language constarined path querying

Navigational queries in edge-labelled graph

- $w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{k-1}} v_k) = l_0 l_1 \ldots l_{k-1}$
- $Q = \{(v_i, v_j) \mid \exists \pi = v_i \to \ldots \to v_j; w(\pi) \in \mathcal{L}\}$, where $\mathcal{L}$ — formal language

- !!!

Applications of Context-Free Path Querying

Static code analysis

Static code analysis

Problems

- There is no unified infrastructure for solutions comparison
  - Data is spread over articles in different communities
  - There is a huge number of different subclasses of the problem
    - all-pairs, single source, multiple source, ...
    - reachability, single path, all path, ...
  - The first and only attempt to compare different algorithms: "An Experimental Study of Context-Free Path Query Evaluation Methods"[1]

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

[2] H. Miao and A. Deshpande, "Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization", 2019

Problems

- There is no unified infrastructure for solutions comparison
  - Data is spread over articles in different communities
  - There is a huge number of different subclasses of the problem
    - all-pairs, single source, multiple source, ...
    - reachability, single path, all path, ...
  - The first and only attempt to compare different algorithms: "An Experimental Study of Context-Free Path Query Evaluation Methods"[1]
  - Conclusion: "We conclude that state of the art solutions are not able to cope with large graphs as found in practice."

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

[2] H. Miao and A. Deshpande, "Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization", 2019

Problems

- There is no unified infrastructure for solutions comparison
  - Data is spread over articles in different communities
  - There is a huge number of different subclasses of the problem
    - all-pairs, single source, multiple source, ...
    - reachability, single path, all path, ...
  - The first and only attempt to compare different algorithms: "An Experimental Study of Context-Free Path Query Evaluation Methods"[1]
  - Conclusion: "We conclude that state of the art solutions are not able to cope with large graphs as found in practice."
- There is no support in real-world graph database
  - H. Miao and A. Deshpande: "Though the problem has been first studied in our community [40], there is little follow up and support in the context of modern graph databases ..."[2]

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

[2] H. Miao and A. Deshpande, "Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization", 2019

Our Results

- ✔ Collection of linear algebra based algorithms for CFPQ
  - SuiteSparse is utilized for sparse linear algebra subroutines
  - Published: `https://github.com/JetBrains-Research/CFPQ_PyAlgo`

[3] "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra", Arseniy Terekhov et al, 2019

Our Results

- ✔ Collection of linear algebra based algorithms for CFPQ
  - SuiteSparse is utilized for sparse linear algebra subroutines
  - Published: `https://github.com/JetBrains-Research/CFPQ_PyAlgo`
- ✔ Full-stack support of CFPQ[3]
  - On the top of RedisGraph
  - openCypher extended to support CFPQ

[3] "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra", Arseniy Terekhov et al, 2019

Our Results

- ✔ Collection of linear algebra based algorithms for CFPQ
  - SuiteSparse is utilized for sparse linear algebra subroutines
  - Published: `https://github.com/JetBrains-Research/CFPQ_PyAlgo`
- ✔ Full-stack support of CFPQ[3]
  - On the top of RedisGraph
  - openCypher extended to support CFPQ
- ⚙ Collecting of the dataset for CFPQ benchmarking is started
  - Synthetic graphs
  - Real-world graphs
    - Static code analysis
    - Biological data analysis
  - Published: `https://github.com/JetBrains-Research/CFPQ_Data`

[3] "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra", Arseniy Terekhov et al, 2019

Our Results Evaluation

- Reachability queries
  - *geospecies* — biological data, folklore
  - *mem* — Andersen points-to analysis
- Time in seconds

- GPGPU: Geforce GTX 1070, 1.5GHz, 8Gb RAM, 1920 CUDA cores
- CPU: Intel core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM

Graph	#V	#E	Neo4j[4]	RedisGraph[5]	Lin.al. CPU[6]	Lin.al. GPGPU[7]
geospecies	450 609	2 311 461	6000.0	80.1	7.1	0.8
Mem 1			n.a.	⚙		
mem 2			n.a.	⚙		
mem 3			n.a.	⚙		

[4] !!!
[5] !!!
[6] !!!
[7] !!!

Ongoing research

- Benchmarking of linear algebra based algorithms
  - Comparison of different algorithms for different query semantics
  - Investigation of scalability on multicore machines
  - Estimation of performance on GPGPU
- GLL-based CFPQ algorithm for Neo4j

Ongoing research

- Benchmarking of linear algebra based algorithms
  - Comparison of different algorithms for different query semantics
  - Investigation of scalability on multicore machines
  - Estimation of performance on GPGPU
- GLL-based CFPQ algorithm for Neo4j
- Semantics of openCypher in terms of linear algebra (in Coq)
- Multiple context-free languages for constraints

Topics to dicuss

- Benchmarks.
  - Algorithms
  - Query language. Semantics
  - Local and Global queries. Can start from RPQ.
  - Queries: templates and real-world queries.
- Competition on language constrained path querying.
  - Part of existing competition for graph processing systems.
  - Involve static analysis community.
- Graph database support
  - Algorithms
  - Query language. Semantics