# gMark: Schema-driven data and workload generation for graph databases

George Fletcher
TU Eindhoven

Joint work with
G. Bagan (Lyon), A. Bonifati (Lyon), R. Ciucanu (Oxford),
A. Lemay (Lille), and N. Advokaat (Eindhoven)

8th LDBC TUC Meeting
Redwood Shores, California
23 June 2016

# Synthetic graph and workload generation with gMark

We present gMark, an open-source framework for generation of synthetic graphs and workloads.

gMark has been designed to tailor diverse graph data management scenarios, which are often driven by query workloads.

# Synthetic graph and workload generation with gMark

We present gMark, an open-source framework for generation of synthetic graphs and workloads.

gMark has been designed to tailor diverse graph data management scenarios, which are often driven by query workloads.

For example

- multi-query optimization,
- mapping discovery and query rewriting in data integration systems,
- workload-driven graph database physical design,

and, in general, flexible specification and generation of diverse workloads addressing particular experimental studies.

# Synthetic graph and workload generation with gMark

Given a graph schema, gMark

- generates synthetic instances of the schema (of desired size)
- generates query workloads with targeted structure and runtime behavior (which holds for all instances of the schema)

# Why gMark?

We adopt successful aspects of the state of the art

For example, like the Waterloo Diversity Benchmark, gMark is schema-driven,

- allowing finely tailored graph instances for specific application domains; and,
- allowing tightly controlled generation of query workloads.

# Why gMark?

We adopt successful aspects of the state of the art

For example, like the Waterloo Diversity Benchmark, gMark is schema-driven,

- allowing finely tailored graph instances for specific application domains; and,
- allowing tightly controlled generation of query workloads.

and, like the LDBC SNB Interactive, gMark supports focused stress-testing of query optimization choke-points, through fine control of query parameters such as selectivities.

# Why gMark?

New features of gMark include

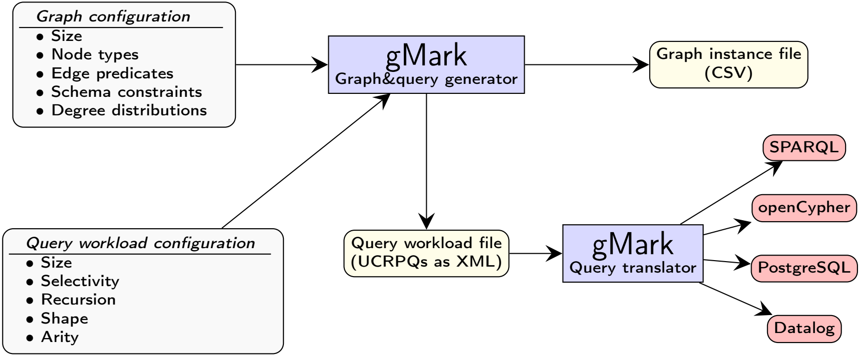
- support for flexible generation of query workloads including recursive path queries, which are fundamental for graph analytics;

# Why gMark?

New features of gMark include

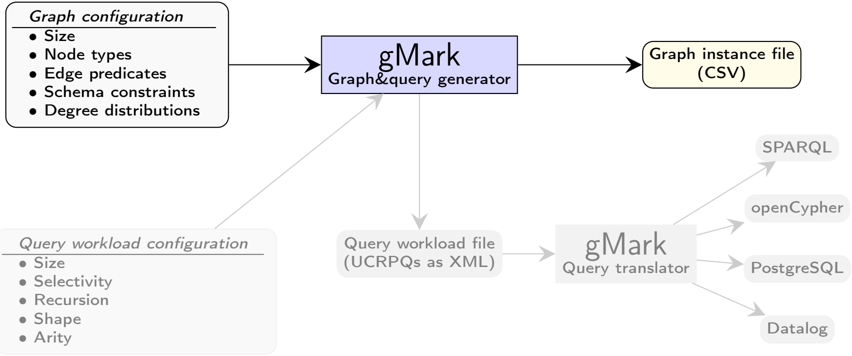
- support for flexible generation of query workloads including recursive path queries, which are fundamental for graph analytics; and,
- query selectivity estimation solution, in a purely instance-independent schema-driven fashion.
  - hence, more scalable, more predictable, and easier to explain/understand.

# Overview of the gMark workflow



# Graph generation

# gMark graph generation



# Graph configurations

The user can specify in the graph configuration (i.e., graph schema):

- **Size**: # of nodes
- **Node types**: finite set of node labels

  e.g., `author`, `citation`, `journal`

# Graph configurations

The user can specify in the graph configuration (i.e., graph schema):

- **Size**: # of nodes
- **Node types**: finite set of node labels

  e.g., `author`, `citation`, `journal`

- **Edge predicates**: finite set of edge labels

  e.g., `authoredBy`, `referencedBy`

# Graph configurations

The user can specify in the graph configuration (i.e., graph schema):

- **Size**: # of nodes
- **Node types**: finite set of node labels

  e.g., `author`, `citation`, `journal`

- **Edge predicates**: finite set of edge labels

  e.g., `authoredBy`, `referencedBy`

- **Schema constraints**: proportion of nodes/edges of given type

  e.g., 20% of all nodes are `authors`

- **Degree distributions**: on the in- and out-degree of edge predicates (uniform, normal, zipfian)

  e.g., the out-distribution of `citation` $\xrightarrow{\texttt{authoredBy}}$ `author` is Gaussian with parameters $\mu = 3, \sigma = 1$

# Graph configurations: Uniprot schema

| *Node type* | *Constr.* |
|---|---|
| gene | 35% |
| protein | 31% |
| author | 20% |
| citation | 10% |
| organism | 1% |
| ... | ... |

**Node types**

| *Edge predicate* | *Constr.* |
|---|---|
| authoredBy | 64% |
| encodedOn | 6% |
| referencedBy | 3% |
| occursIn | 2% |
| ... | ... |

**Edge predicates**

| *source type* $\underrightarrow{\textit{predicate}}$ *target type* | *In-distr.* | *Out-distr.* |
|---|---|---|
| protein $\underrightarrow{\texttt{encodedOn}}$ gene | Zipfian | Gaussian |
| protein $\underrightarrow{\texttt{occursIn}}$ organism | Zipfian | Uniform |
| protein $\underrightarrow{\texttt{referencedBy}}$ citation | Zipfian | Gaussian |
| citation $\underrightarrow{\texttt{authoredBy}}$ author | Zipfian | Gaussian |
| ... | ... | ... |

**In- and out-degree distributions**

# Schema-driven graph generation

We have established the intractability of the generation problem

**Theorem**

*Given a graph configuration $G$, deciding whether or not there exists a graph instance satisfying $G$ is NP-complete.*

# Schema-driven graph generation

We have established the intractability of the generation problem

**Theorem**

*Given a graph configuration $G$, deciding whether or not there exists a graph instance satisfying $G$ is NP-complete.*

Hence, gMark follows a 'best-effort' strategy in instance generation, i.e., it attempts to achieve the exact values of the input parameters and relaxes them whenever this is not possible.

# Schema-driven graph generation

We have adapted the scenarios of several popular use cases into meaningful gMark configurations, while also adding new gMark features:

- `Bib`: our default bibliographical use-case
- `LSN`: LDBC social network benchmark
- `WD`: WatDiv e-commerce benchmark
- `SP`: SP2Bench DBLP benchmark

# Scalability of gMark graph generation

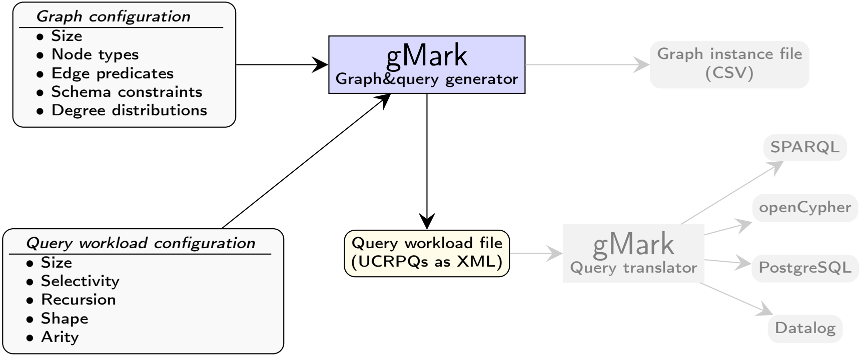
| | 100K | 1M | 10M | 100M |
|---|---|---|---|---|
| `Bib` | 0m0.057s | 0m0.638s | 0m8.344s | 1m28.725s |
| `LSN` | 0m0.225s | 0m1.451s | 0m23.018s | 3m11.318s |
| `WD` | 0m2.163s | 0m25.032s | 4m10.988s | 113m31.078s |
| `SP` | 0m0.638s | 0m7.048s | 1m28.831s | 15m23.542s |

*Graph generation times, with varying graph sizes (# nodes)*

Generation time depends heavily on density of instances (e.g., `WD` has 100x number of edges than `Bib`)

# Query workload generation

# gMark query generation



# A query language for graphs

UCRPQ: Unions of Conjunctions of Regular Path Queries
– Core constructs of the W3C's SPARQL 1.1, Oracle's PGQL, and and Neo4j's openCypher
– Well understood theoretical properties (e.g., polynomial data complexity)

# A query language for graphs

UCRPQ: Unions of Conjunctions of Regular Path Queries
– Core constructs of the W3C's SPARQL 1.1, Oracle's PGQL, and and Neo4j's openCypher
– Well understood theoretical properties (e.g., polynomial data complexity)

UCRPQ includes **recursive queries** (via the Kleene star *), with applications in social networks, bioinformatics, etc.

gMark generates UCRPQ → the first schema-driven tool to support recursive queries and their generation in concrete syntaxes.

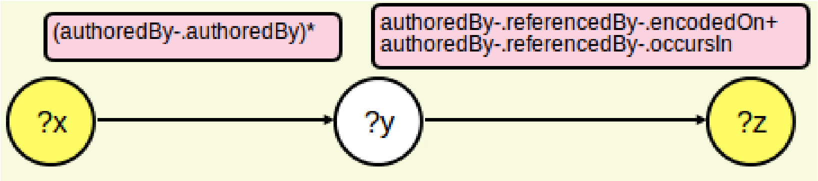
# A query language for graphs

Example of UCRPQ

*for each researcher, select all of the biological entities (i.e., genes and organisms) relevant to proteins studied in papers authored by people in the researcher's coauthorship network*

Example of UCRPQ

*for each researcher, select all of the biological entities (i.e., genes and organisms) relevant to proteins studied in papers authored by people in the researcher's coauthorship network*



# A query language for graphs

Example of UCRPQ

*for each researcher, select all of the biological entities (i.e., genes and organisms) relevant to proteins studied in papers authored by people in the researcher's coauthorship network*

$$\boxed{(?x, ?z) \leftarrow (?x, (\mathtt{a}^{-} \cdot \mathtt{a})^{*}, ?y), (?y, (\mathtt{a}^{-} \cdot \mathtt{r}^{-} \cdot \mathtt{e} + \mathtt{a}^{-} \cdot \mathtt{r}^{-} \cdot \mathtt{o}), ?z)}$$

$\big($a=authoredBy, r=referencedBy, e=encodedOn, o=occursIn$\big)$

# A query language for graphs

Example of UCRPQ

*for each researcher, select all of the biological entities (i.e., genes and organisms) relevant to proteins studied in papers authored by people in the researcher's coauthorship network*

$$\boxed{(?x, ?z) \leftarrow (?x, (\mathtt{a}^{-} \cdot \mathtt{a})^{*}, ?y), (?y, (\mathtt{a}^{-} \cdot \mathtt{r}^{-} \cdot \mathtt{e} + \mathtt{a}^{-} \cdot \mathtt{r}^{-} \cdot \mathtt{o}), ?z)}$$

$\left(\mathtt{a}{=}\mathtt{authoredBy},\ \mathtt{r}{=}\mathtt{referencedBy},\ \mathtt{e}{=}\mathtt{encodedOn},\ \mathtt{o}{=}\mathtt{occursIn}\right)$

| | |
|---|---|
| #rules | 1 |
| #conjuncts | 2 |
| #disjuncts | 1, 2 |
| path lengh | 2, 3, 3 |

# Schema-driven workload generation

The user can specify in the query workload configuration:

- **Size**: #queries, #conjuncts/#disjuncts/path length per query
- **Selectivity**: constant, linear, quadratic.
- **Recursion**: probability to generate Kleene star above a conjunct.

# Schema-driven workload generation

The user can specify in the query workload configuration:

- **Size**: #queries, #conjuncts/#disjuncts/path length per query
- **Selectivity**: constant, linear, quadratic.
- **Recursion**: probability to generate Kleene star above a conjunct.
- **Shape**: chain, star, cycle, star-chain.
- **Arity**: arbitrary (including 0 i.e., Boolean).

The graph configuration is also input to the query generator.

# Schema-driven workload generation

Approach

1. Prepare “schema” and “selectivity” graphs

# Schema-driven workload generation

## Approach

1. Prepare "schema" and "selectivity" graphs
2. For each query to be generated:
   - 2.1 Generate query skeleton
   - 2.2 Assign selectivity class to each predicate
   - 2.3 Instantiate each predicate

# Schema-driven workload generation

Approach

1. Prepare “schema” and “selectivity” graphs
2. For each query to be generated:
   - 2.1 Generate query skeleton
   - 2.2 Assign selectivity class to each predicate
   - 2.3 Instantiate each predicate

Assigning selectivities required us to develop a non-trivial infrastructure for instance-independent reasoning over query behavior, based on a Selectivity Algebra.

# Selectivity estimation quality of gMark

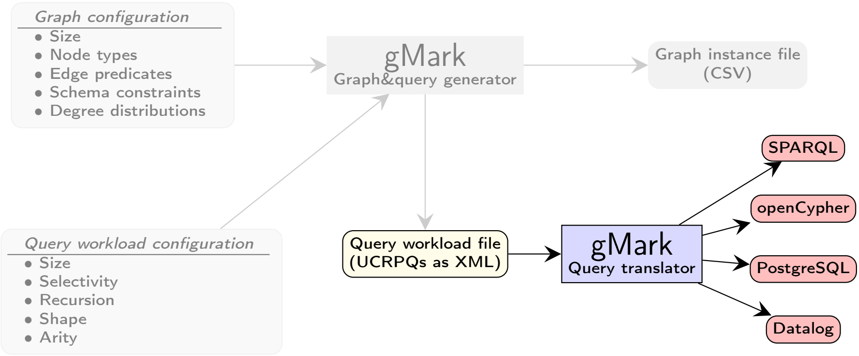
- Given a binary query $Q$ and a graph $G$, we assume that $|Q(G)| = \mathcal{O}(|nodes(G)|^{\alpha})$.
- $\alpha$ is the selectivity value (0–constant, 1–linear, 2–quadratic).

# Selectivity estimation quality of gMark

- Given a binary query $Q$ and a graph $G$, we assume that $|Q(G)| = \mathcal{O}(|nodes(G)|^{\alpha})$.
- $\alpha$ is the selectivity value (0–constant, 1–linear, 2–quadratic).
- Experiments confirmed the assumption and the estimation quality.

| | *Constant* | *Linear* | *Quadratic* |
|---|---|---|---|
| `LSN`-*Len* | 0.200±0.417 | 1.189±0.261 | 2.032±0.059 |
| `LSN`-*Dis* | 0.182±0.364 | 1.325±0.318 | 2.046±0.074 |
| `LSN`-*Con* | 0.190±0.391 | 1.244±0.326 | 2.017±0.032 |
| `LSN`-*Rec* | 0.196±0.409 | 1.090±0.492 | 1.564±0.889 |
| `Bib`-*Len* | 0.003±0.010 | 0.921±0.122 | 1.405±0.337 |
| `Bib`-*Dis* | 0.000±0.000 | 0.995±0.012 | 1.607±0.261 |
| `Bib`-*Con* | 0.023±0.029 | 0.986±0.112 | 1.409±0.296 |
| `Bib`-*Rec* | 0.100±0.316 | 0.982±0.073 | 1.493±0.335 |
| `WD`-*Len* | 0.016±0.044 | 1.427±0.392 | 2.004±0.022 |
| `WD`-*Dis* | 0.009±0.022 | 1.412±0.380 | 1.999±0.014 |
| `WD`-*Con* | -0.010±0.026 | 1.540±0.495 | 1.750±0.708 |
| `WD`-*Rec* | 0.587±0.830 | - | 1.976±0.012 |
| `SP` | 0.074±0.130 | 1.064±0.034 | 2.034±0.295 |

# gMark query translator



# Query translation

$\boxed{\text{UCRPQ: } (?x, ?z) \leftarrow (?x, (\mathtt{a}^{-} \cdot \mathtt{a})^{*}, ?y), (?y, (\mathtt{a}^{-} \cdot \mathtt{r}^{-} \cdot \mathtt{e} + \mathtt{a}^{-} \cdot \mathtt{r}^{-} \cdot \mathtt{o}), ?z)}$

# Query translation

UCRPQ: $(?x, ?z) \leftarrow (?x, (\mathtt{a}^- \cdot \mathtt{a})^*, ?y), (?y, (\mathtt{a}^- \cdot \mathtt{r}^- \cdot \mathtt{e} + \mathtt{a}^- \cdot \mathtt{r}^- \cdot \mathtt{o}), ?z)$

**SPARQL**

```
PREFIX : <http://example.org/gmark/>
SELECT DISTINCT ?x ?z
WHERE { ?x (^:a/:a)* ?y .
?y ((^:a/^:r/:e)|(^:a/^:r/:o)) ?z .}
```

**openCypher**

```
MATCH (x)<-[:a]-()-[:a]->(y),
(y)<-[:a]-()<-[:r]-()-[:e]->(z)
RETURN DISTINCT x, z
UNION
MATCH (x)<-[:a]-()-[:a]->(y),
(y)<-[:a]-()<-[:r]-()-[:o]->(z)
RETURN DISTINCT x, z;
```

**Datalog**

```
g0(x,y)<- edge(x1,a,x0),edge(x1,a,x2),
   x=x0,y=x2.
g0(x,y)<- g0(x,z),g0(z,y).
g1(x,y)<- edge(x1,a,x0),edge(x2,r,x1),
   edge(x2,e,x3),x=x0,y=x3.
g1(x,y)<- edge(x1,a,x0),edge(x2,r,x1),
   edge(x2,o,x3),x=x0,y=x3.
query(x,z)<- g0(x,y),g1(y,z).
```

**SQL**

```
WITH RECURSIVE c0(src, trg) AS (
SELECT edge.src, edge.src FROM edge
UNION
SELECT edge.trg, edge.trg FROM edge
UNION
SELECT s0.src, s0.trg
FROM (SELECT trg as src, src as trg,
.........................................
```

# Scalability of gMark workload generation

On my laptop, gMark easily generates workloads of one thousand queries for `Bib` in $\sim 0.3s$; `LSN` and `SP` in $\sim 1.5s$; and for the richer `WD` scenario in $\sim 10s$.

Query translation of the thousand queries into all four supported syntaxes for each of the four scenarios required $\sim 0.1s$.

# Scalability on recursive query workloads

Example Application. We performed an extensive performance study of four state-of-the-art systems under the four use-case schemas.

Our main finding was that performance on queries containing recursive path navigation (i.e., RPQs) was typically impractical

- indicates the need for further study of the engineering of this basic class of graph queries

# Wrap Up

# Recap

Novel contributions of gMark

# Recap

Novel contributions of gMark

- schema-driven graph and query-workload generation, featuring instance-independent selectivity estimation;

# Recap

Novel contributions of gMark

- schema-driven graph and query-workload generation, featuring instance-independent selectivity estimation;
- finely controlled query workload-centered approach
  - versus query-centered approaches – nb. both are valid and needed!

# Recap

Novel contributions of gMark

- schema-driven graph and query-workload generation, featuring instance-independent selectivity estimation;
- finely controlled query workload-centered approach
  - versus query-centered approaches – nb. both are valid and needed!
- discovery of the performance difficulties of existing graph DBMS's on evaluating a basic class of graph queries
  - Regular Path Queries

# Recap

Novel contributions of gMark

- schema-driven graph and query-workload generation, featuring instance-independent selectivity estimation;
- finely controlled query workload-centered approach
  - versus query-centered approaches – nb. both are valid and needed!
- discovery of the performance difficulties of existing graph DBMS's on evaluating a basic class of graph queries
  - Regular Path Queries

Come see us at our **VLDB 2016 demo**!

# Looking ahead to gMark v2.0

To-do/wishlist.

- richer queries
  - support of constants in queries
  - additional query shapes
  - aggregation for BI workloads
  - extensions of selectivity estimation to higher arity queries,

# Looking ahead to gMark v2.0

To-do/wishlist.

- richer queries
  - support of constants in queries
  - additional query shapes
  - aggregation for BI workloads
  - extensions of selectivity estimation to higher arity queries,
- richer schemas
  - configuration parameter completion,
  - schema constructs for correlated structure,
  - class hierarchies

# Looking ahead to gMark v2.0

To-do/wishlist.

- richer queries
  - support of constants in queries
  - additional query shapes
  - aggregation for BI workloads
  - extensions of selectivity estimation to higher arity queries,
- richer schemas
  - configuration parameter completion,
  - schema constructs for correlated structure,
  - class hierarchies
- align our work with LDBC activites?

# gMark: Schema-driven data and workload generation for graph databases


George Fletcher

TU Eindhoven, Netherlands

Joint work with

G. Bagan (Lyon), A. Bonifati (Lyon), R. Ciucanu (Oxford), A. Lemay (Lille), and N. Advokaat (Eindhoven)


`https://github.com/graphMark/gmark`

Thank you!