Fine-Grained Complexity of Regular Path Queries

Katrin Casel¹, Markus L. Schmid²

¹ HPI, University of Potsdam, Germany
² HU Berlin, Germany

ICDT 2021
Querying Graphs with Regular Expressions

Graph databases

 directed, edge-labelled multigraphs.
Querying Graphs with Regular Expressions

**Graph databases**
directed, edge-labelled multigraphs.

| **Graph Databases** | **Expression** |
|---------------------|---------------|
| $\Sigma$            | finite alphabet (edge labels) |
| $V_D$               | vertices (or nodes) |
| $E_D \subseteq V_D \times \Sigma \times V_D$ | edges (or arcs) |
| Graph database      | $D = (V_D, E_D)$ |
Graph databases

Directed, edge-labelled multigraphs.

Graph Databases

\[
\begin{align*}
\Sigma & \quad \text{finite alphabet (edge labels)} \\
V_D & \quad \text{vertices (or nodes)} \\
E_D & \subseteq V_D \times \Sigma \times V_D \quad \text{edges (or arcs)} \\
\text{Graph database} & \quad \mathcal{D} = (V_D, E_D)
\end{align*}
\]

Regular Path Queries (RPQs)

Regular expressions \( q \) over \( \Sigma \).

\[ q(\mathcal{D}) = \{ (u, v) \mid \exists \ u\text{-to-}v \text{ path labelled by a word from } \mathcal{L}(q) \} \]
Regular Path Query Example

Graph database $\mathcal{D}$:
Regular Path Query Example

Graph database $\mathcal{D}$:

Regular path query:

$$q = a^*(b \lor c)$$
Different Variants of RPQs
Different Variants of RPQs

Query results:
- Only node pairs \((u, v)\).
- Node pairs \((u, v)\) and a witness path.
- Node pairs \((u, v)\) and all witness paths.
Different Variants of RPQs

Query results:
- Only node pairs \((u, v)\).
- Node pairs \((u, v)\) and a witness path.
- Node pairs \((u, v)\) and all witness paths.

Path semantics: \((u, v) \in q(D)\) if there is
- an arbitrary path.
- a simple path.
- a trail.
- a shortest path.
Product Graph Approach (PG-Approach)

\[ D: \text{Graph database} \]
\[ q: \text{Regular path query} \]
\[ M: \text{NFA for } q \text{ with state set } Q \]
Product Graph Approach (PG-Approach)

| $\mathcal{D}$: | Graph database |
|----------------|----------------|
| $q$: | Regular path query |
| $M$: | NFA for $q$ with state set $Q$ |

**Product Graph**

\[
G(\mathcal{D}, q) = (V(\mathcal{D}, q), E(\mathcal{D}, q))
\]

\[
V(\mathcal{D}, q) = V_\mathcal{D} \times Q
\]

\[
E(\mathcal{D}, q) \subseteq (V(\mathcal{D}, q) \times V(\mathcal{D}, q)):
\]

\[
(u, p) \rightarrow (v, p') \iff \exists x \in \Sigma : u \xrightarrow{x} v \land p \xrightarrow{x} p'.
\]
PG-Approach Example
PG-Approach Example
## RPQ Evaluation Tasks

| Name           | Input   | Task                                           |
|----------------|---------|------------------------------------------------|
| RPQ-Boole      | $D, q$  | Decide whether $q(D) = \emptyset$.             |
| RPQ-Eval       | $D, q$  | Compute the whole set $q(D)$.                  |
| RPQ-Count      | $D, q$  | Compute $|q(D)|$.                             |
| (Sorted) RPQ-Enum | $D, q$ | Enumerate the whole set $q(D)$ (lexicographically ordered). |
## RPQ Evaluation Tasks

| Name                  | Input | Task                                                      |
|-----------------------|-------|-----------------------------------------------------------|
| RPQ-Boole             | $D, q$| Decide whether $q(D) = \emptyset$.                        |
| RPQ-Eval              | $D, q$| Compute the whole set $q(D)$.                             |
| RPQ-Count             | $D, q$| Compute $|q(D)|$.                                         |
| (Sorted) RPQ-Enum     | $D, q$| Enumerate the whole set $q(D)$ (lexicographically ordered).|

Updates: Adding/deleting isolated nodes, adding/deleting arcs.
Research Question

- PG-approach good for simple tasks like checking $q(D) = \emptyset$ or $(u, v) \in q(D)$.
  What about computing, counting or enumerating $q(D)$?
- Is the PG-approach optimal?
- Can we complement upper bounds with conditional lower bounds?
Fine-Grained Complexity and Conditional Lower Bounds
Orthogonal Vectors (OV)

Input: Sets $A, B$ each containing $n$ Boolean $d$-dimensional vectors.

Question: Are there orthogonal vectors $\vec{a} \in A$ and $\vec{b} \in B$?
Orthogonal Vectors (OV)

Input: Sets $A, B$ each containing $n$ Boolean $d$-dimensional vectors.
Question: Are there orthogonal vectors $\vec{a} \in A$ and $\vec{b} \in B$?

OV-Hypothesis

For every $\epsilon > 0$, OV cannot be solved in $O(n^{2-\epsilon} \text{poly}(d))$. 
Boolean Matrix Multiplication

Boolean Matrix Multiplication (BMM)

Input: Boolean $n \times n$ matrices $A, B$.
Task: Compute $A \times B$. 
### Boolean Matrix Multiplication (BMM)

**Input:** Boolean $n \times n$ matrices $A, B$.

**Task:** Compute $A \times B$.

### com-BMM-Hypothesis

For every $\epsilon > 0$, BMM cannot be solved in $O(n^{3-\epsilon})$ by a combinatorial algorithm.
| Boolean Matrix Multiplication (BMM) |
|-------------------------------------|
| Input: Boolean $n \times n$ matrices $A, B$. |
| Task: Compute $A \times B$. |

| com-BMM-Hypothesis |
|---------------------|
| For every $\epsilon > 0$, BMM cannot be solved in $O(n^{3-\epsilon})$ by a combinatorial algorithm. |

| SBMM-Hypothesis |
|-----------------|
| BMM cannot be solved in $O(m)$, where $m =$ number of 1-entries. |
Our Results
Theorem

RPQ-Boole can be solved in time $O(|D||q|)$. 
**Theorem**

RPQ-Boole can be solved in time $O(|D| |q|)$.

| Theorem |
|----------------|
| **If RPQ-Boole can be solved in time** |
| ★ $O(|D|^{2-\epsilon} + |q|^{2})$, then OV-hypothesis fails. |
| ★ $O(|D|^{2} + |q|^{2-\epsilon})$, then OV-hypothesis fails. |
| ★ $O(|V_D|^{3-\epsilon} + |q|^{3-\epsilon})$, com-BMM-hypothesis fails. |
**Theorem**

RPQ-Boole can be solved in time $O(|D||q|)$.

**Theorem**

If RPQ-Boole can be solved in time

- $O(|D|^{2-\epsilon} + |q|^2)$, then OV-hypothesis fails.
- $O(|D|^2 + |q|^{2-\epsilon})$, then OV-hypothesis fails.
- $O(|V_D|^{3-\epsilon} + |q|^{3-\epsilon})$, com-BMM-hypothesis fails.

**Data Complexity**

From now on ALL bounds in data complexity!
RPQ-Eval and RPQ-Count

| Theorem |
|------------------|
| RPQ-Eval (and RPQ-Count) can be solved in time \(O(|V_D||D|)\). |
### Theorem

**RPQ-Eval and RPQ-Count**

**Theorem**

RPQ-Eval (and RPQ-Count) can be solved in time $O(|V_D||D|)$.

**Theorem**

If RPQ-Eval can be solved in time

- $O((|V_D||D|)^{1-\epsilon})$, then com-BMM-hypothesis fails.
- $O((|q(D)| + |D|))$, then SBMM-hypothesis fails.
## RPQ-Eval and RPQ-Count

| Statement |
|-----------|
| **Theorem** |
| RPQ-Eval (and RPQ-Count) can be solved in time $O(|V_D||D|)$. |

| Statement |
|-----------|
| **Theorem** |
| If RPQ-Eval can be solved in time $O((|V_D||D|)^{1-\epsilon})$, then com-BMM-hypothesis fails. |
| If RPQ-Eval can be solved in time $O((|q(D)| + |D|))$, then SBMM-hypothesis fails. |

| Statement |
|-----------|
| **Theorem** |
| If RPQ-Count can be solved in time $O((|V_D||D|)^{1-\epsilon})$ then the OV-hypothesis fails. |
Theorem

Sorted RPQ-Enum can be solved with preprocessing $O(|\mathcal{D}|)$, delay $O(|\mathcal{D}|)$ and $O(1)$ updates.
Theorem

Sorted RPQ-Enum can be solved with preprocessing $O(|D|)$, delay $O(|D|)$ and $O(1)$ updates.

Some Thoughts

- Linear preprocessing is reasonable.
- Linear delay is bad.
- What about updates??
RPQ-Enum – Lower Bounds

Conditional Lower Bounds

Linear preprocessing and

► constant delay? No!
Conditional Lower Bounds

Linear preprocessing and
  ▶ constant delay? No!
  ▶ delay sublinear in $|V_D|$? No!
## Conditional Lower Bounds

Linear preprocessing and

- constant delay? No!
- delay sublinear in $|V_D|$? No!
- delay sublinear in $|\mathcal{D}|$? Not if we also want updates!
### Conditional Lower Bounds

Linear preprocessing and
- constant delay? No!
- delay sublinear in $|V_D|$? No!
- delay sublinear in $|D|$? Not if we also want updates!

### Open Question

RPQ-Enum with $O(|D|)$ preprocessing and $O(|V_D|)$ delay???
RPQ-Enum – Lower Bounds

Conditional Lower Bounds

Linear preprocessing and

- constant delay? No!
- delay sublinear in $|V_D|$? No!
- delay sublinear in $|D|$? Not if we also want updates!

Open Question

RPQ-Enum with $O(|D|)$ preprocessing and $O(|V_D|)$ delay???

Next objective:

Just any enumeration that guarantees delay sublinear in $|D|$.
### Three Approaches to Sublinear Delay

| First Approach: Representative Subset of Solution Set |
|-----------------------------------------------------|
| A “representative” subset $A \subseteq q(D)$ can be enumerated with linear preprocessing and constant delay. |
Three Approaches to Sublinear Delay

First Approach: Representative Subset of Solution Set

A “representative” subset $A \subseteq q(D)$ can be enumerated with linear preprocessing and constant delay.

$\Delta(D)$ denotes the average degree of $D$.

Second Approach: Super-Linear Preprocessing

Sorted RPQ-Enum can be solved with preprocessing $O(\log(\Delta(D)) \Delta(D) |D|)$ and delay $O(|V_D|)$.
Three Approaches to Sublinear Delay

First Approach: Representative Subset of Solution Set

A “representative” subset $A \subseteq q(D)$ can be enumerated with linear preprocessing and constant delay.

$\Delta(D)$ denotes the average degree of $D$.

Second Approach: Super-Linear Preprocessing

Sorted RPQ-Enum can be solved with preprocessing $O(\log(\Delta(D)) \Delta(D) |D|)$ and delay $O(|V_D|)$.

$\Delta(D)$ denotes the maximum degree of $D$.

Third Approach: Restricted Class of RPQs

For a $Q \subseteq \text{RPQ}$, RPQ-Enum can be solved with preprocessing $O(|D|)$ and delay $O(\Delta(D))$. 
Third Approach: Restricted Class of RPQs

- **Short RPQ (S-RPQ):**
  
  \[ q = (x_1 \lor \ldots \lor x_k) \text{ or } q = (x_1 \lor \ldots \lor x_k)(y_1 \lor \ldots \lor y_{k'}) \]

  where \( x_1, \ldots, x_k, y_1, \ldots, y_{k'} \in \Sigma \).

  Example: \( q = (a \lor b)(a \lor c \lor d) \).
Third Approach: Restricted Class of RPQs

- **Short RPQ (S-RPQ):**
  \[ q = (x_1 \lor \ldots \lor x_k) \text{ or } q = (x_1 \lor \ldots \lor x_k)(y_1 \lor \ldots \lor y_{k'}), \]
  where \( x_1, \ldots, x_k, y_1, \ldots, y_{k'} \in \Sigma. \)
  
  Example: \( q = (a \lor b)(a \lor c \lor d). \)

- **Basic Transitive RPQ (BT-RPQ):**
  \[ q = (x_1 \lor \ldots \lor x_k)^* \text{ or } q = (x_1 \lor \ldots \lor x_k)^+, \]
  where \( x_1, \ldots, x_k \in \Sigma. \)
  
  Example: \( q = (a \lor c \lor d)^+. \)
Third Approach: Restricted Class of RPQs

- **Short RPQ (S-RPQ):**
  \[ q = (x_1 \lor \ldots \lor x_k) \text{ or } q = (x_1 \lor \ldots \lor x_k)(y_1 \lor \ldots \lor y_{k'}) \]
  where \( x_1, \ldots, x_k, y_1, \ldots, y_{k'} \in \Sigma \).
  
  Example: \( q = (a \lor b)(a \lor c \lor d) \).

- **Basic Transitive RPQ (BT-RPQ):**
  \[ q = (x_1 \lor \ldots \lor x_k)^* \text{ or } q = (x_1 \lor \ldots \lor x_k)^+, \]
  where \( x_1, \ldots, x_k \in \Sigma \).
  
  Example: \( q = (a \lor c \lor d)^+ \).

- **Alternation Closure:**
  \[ \lor(S-RPQ \cup BT-RPQ) = \]
  \[ \{(q_1 \lor \ldots \lor q_m) \mid q_i \in S-RPQ \cup BT-RPQ, 1 \leq i \leq m\} \]
  
  Example: \( q = (ab \lor c^* \lor b(c \lor d) \lor (a \lor b \lor d)^+) \)
Theorem

Semi-sorted Enum(\(\bigvee (S\text{-RPQ} \cup BT\text{-RPQ})\)) can be solved with preprocessing \(O(|\mathcal{D}|)\) and delay \(O(\Delta(\mathcal{D}))\).
Third Approach: Restricted Class of RPQs

Theorem

Semi-sorted Enum(\(\bigvee (S-RPQ \cup BT-RPQ)\)) can be solved with preprocessing \(O(|D|)\) and delay \(O(\Delta(D))\).

Proof Sketch

- *Semi-sorted* Enum(S-RPQ) and Enum(BT-RPQ) can be solved with preprocessing \(O(|D|)\) and delay \(O(\Delta(D))\).
### Third Approach: Restricted Class of RPQs

| Theorem |
|-----------------------------|
| Semi-sorted Enum(\(\bigvee (S\text{-RPQ} \cup BT\text{-RPQ})\)) can be solved with preprocessing \(O(|D|)\) and delay \(O(\Delta(D))\). |

| Proof Sketch |
|-------------------------------------|
| ▶ *Semi-sorted* Enum(S-RPQ) and Enum(BT-RPQ) can be solved with preprocessing \(O(|D|)\) and delay \(O(\Delta(D))\). |
| ▶ For every \(Q \subseteq \text{RPQ}:\) Semi-sorted Enum(\(Q\)) can be solved with linear preprocessing and some delay, then Enum(\(\bigvee (Q)\)) can be solved with the same preprocessing and delay. |
## Third Approach: Restricted Class of RPQs

### Theorem

Semi-sorted Enum(\(\bigvee (S\text{-RPQ} \cup BT\text{-RPQ})\)) can be solved with preprocessing \(O(|D|)\) and delay \(O(\Delta(D))\).

### Proof Sketch

- **Semi-sorted** Enum(S-RPQ) and Enum(BT-RPQ) can be solved with preprocessing \(O(|D|)\) and delay \(O(\Delta(D))\).
- For every \(Q \subseteq RPQ\): Semi-sorted Enum(\(Q\)) can be solved with linear preprocessing and some delay, then Enum(\(\bigvee (Q)\)) can be solved with the same preprocessing and delay.

\(\square\)

### Theorem

If RPQ-Enum(S-RPQ) can be solved with preprocessing \(O(|V_D|^{3-\epsilon})\) and delay \(O(|\Delta(D)|^{1-\epsilon})\), then the com-BMM-hypothesis fails.
Thank you very much for your attention.