Constant delay enumeration for FO queries over nowhere dense graphs

Nicole Schweikardt\textsuperscript{1}, Luc Segoufin\textsuperscript{2} and Alexandre Vigny\textsuperscript{3}

\textsuperscript{1}Humboldt-Universität zu Berlin, Berlin

\textsuperscript{2}Inria, ENS Ulm, Paris

\textsuperscript{3}Université Paris Diderot, Paris

PODS, June 11, 2018
Query evaluation

- Query \( q \)
- Database \( D \)
- Compute \( q(D) \)

Examples:

query \( q \)
first order logic

\[
q(x, y) := \exists z (B(x) \land E(x, z) \land \neg E(y, z))
\]

database \( D \)
relational structure

solutions \( q(D) \)
set of tuples

\[
\{(1,2), (1,3), (1,4), (1,6), (1,7), \ldots, (3,1), (3,2), (3,4), (3,6), (3,7), \ldots\}
\]
Too many solutions!

Database: A given store that contains 50 items for less than 1€
Query: What can I buy with 10€?
Solutions: At least $50^{10}$ possibilities!

- For practical reasons:
  A set of $50^{10}$ solutions is not easy to store / display!

- For theoretical reasons:
  The time needed to compute the answer does not reflect the hardness of the problem.
**Enumeration**

Input: \( \|D\| := n \quad \& \quad |q| := k \)  
(computation with RAM)

Goal: output solutions one by one  
(no repetition)

---

- **STEP 1: Preprocessing**

  Prepare the enumeration: Database \( D \rightarrow I \)

  Preprocessing time: \( f(k) \cdot n \sim O(n) \)

- **STEP 2: Enumeration**

  Enumerate the solutions: Index \( I \rightarrow x_1, x_2, x_3, x_4, \ldots \)

  Delay: \( O(f(k)) \sim O(1) \)

**Constant delay enumeration after linear preprocessing**  
\( (O(1) \circ O(n)) \)
Example 1

Input:
- Database $D := \langle \{1, \cdots, n\}; E \rangle$ \quad $\|D\| = |E|$
- Query $q(x, y) := \neg E(x, y)$

$D$

\[
\begin{array}{ll}
(1,1) & \\
(1,2) & \\
(1,6) & \\
\vdots & \\
(2,3) & \\
\vdots & \\
(i,j) & \\
(i,j+1) & \\
(i,j+3) & \\
\vdots & \\
(n,n) & \\
\end{array}
\]
Example 1

Input:
- Database $D := \langle \{1, \cdots, n\}; E \rangle \quad \|D\| = |E|$
- Query $q(x, y) := \neg E(x, y)$

\[
\begin{array}{c|c}
\text{D} & \text{Index} \\
\hline
(1,1) & (1,1) \\
(1,2) & (1,2) \rightarrow (1,3) \\
(1,6) & \vdots \rightarrow (2,4) \\
(2,3) & \vdots \\
(i,j) & (i,j) \rightarrow (i,j+2) \\
(i,j+1) & (i,j+1) \rightarrow (i,j+2) \\
(i,j+3) & (i,j+3) \rightarrow (k,l) \\
\vdots & \vdots \\
(n,n) & (n,n) \rightarrow \text{NULL}
\end{array}
\]
Example 1

Input:
- Database $D := \langle\{1, \ldots, n\}; E\rangle$, $\|D\| = |E|$
- Query $q(x, y) := \neg E(x, y)$

<table>
<thead>
<tr>
<th>D</th>
<th>Index</th>
<th>Enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>(1,1)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(1,2)</td>
<td>(1,2)</td>
<td>(1,3)</td>
</tr>
<tr>
<td>(1,6)</td>
<td>(1,6)</td>
<td>(1,5)</td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>(2,3)</td>
<td>(2,3)</td>
<td>(2,4)</td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>(i,j)</td>
<td>(i,j)</td>
<td>(i,j+2)</td>
</tr>
<tr>
<td>(i,j+1)</td>
<td>(i,j+1)</td>
<td>(i,j+2)</td>
</tr>
<tr>
<td>(i,j+3)</td>
<td>(i,j+3)</td>
<td>(k,l)</td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>(n,n)</td>
<td>(n,n)</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Alexandre Vigny
Enumeration & nowhere dense graphs
PODS, June 11, 2018 5 / 18
Example 2

Input:
- Database $D := \langle\{1, \cdots, n\}; E_1; E_2\rangle$ \quad $\|D\| = |E_1| + |E_2|$ \quad ($E_i \subseteq D \times D$)
- Query $q(x, y) := \exists z, \ E_1(x, z) \land E_2(z, y)$
Example 2

Input:
- Database \( D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle \) \( \|D\| = |E_1| + |E_2| \) \( (E_i \subseteq D \times D) \)
- Query \( q(x,y) := \exists z, E_1(x,z) \land E_2(z,y) \)

\[ A : \text{Adjacency matrix of } E_1 \]
\[ B : \text{Adjacency matrix of } E_2 \]
\[ C : \text{Result matrix} \]
Example 2

Input:
- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle$ \quad $\|D\| = |E_1| + |E_2|$ \quad ($E_i \subseteq D \times D$)
- Query $q(x, y) := \exists z, E_1(x, z) \land E_2(z, y)$

$B$ : Adjacency matrix of $E_2$

$$
\begin{pmatrix}
E_2(1,1) & \cdots & E_2(1, y) & \cdots & E_2(1, n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
E_2(z, 1) & \cdots & E_2(z, y) & \cdots & E_2(z, n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
E_2(n, 1) & \cdots & E_2(n, y) & \cdots & E_2(n, n)
\end{pmatrix}
$$

Compute the set of solutions

$=$

boolean matrix multiplication

$A$ : Adjacency matrix of $E_1$

$$
\begin{pmatrix}
E_1(1,1) & \cdots & E_1(1, i) & \cdots & E_1(1, n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
E_1(x, 1) & \cdots & E_1(x, z) & \cdots & E_1(x, n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
E_1(n, 1) & \cdots & E_1(n, z) & \cdots & E_1(n, n)
\end{pmatrix}
$$

$C$ : Result matrix

$$
\begin{pmatrix}
q(1,1) & \cdots & q(1, y) & \cdots & q(1, n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
q(x, 1) & \cdots & q(x, y) & \cdots & q(x, n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
q(n, 1) & \cdots & q(n, y) & \cdots & q(n, n)
\end{pmatrix}
$$
Example 2

Input:
- Database $D := \langle\{1, \cdots, n\}; E_1; E_2\rangle$ \quad $\|D\| = |E_1| + |E_2|$ \quad ($E_i \subseteq D \times D$)
- Query $q(x, y) := \exists z, E_1(x, z) \land E_2(z, y)$

If we enumerate that efficiently:
- Linear preprocessing: $O(n^2)$
- Number of solutions: $O(n^2)$
- Algorithm for the boolean matrix multiplication in $O(n^2)$

Conjecture: "There are no algorithm for the boolean matrix multiplication working in time $O(n^2)$."
Example 2

Input:
- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle$  \quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)
- Query $q(x, y) := \exists z, \ E_1(x, z) \land E_2(z, y)$

This query cannot be enumerated with constant delay\textsuperscript{1}

\textsuperscript{1}Unless there is a breakthrough with the boolean matrix multiplication.
Example 2

Input:
- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle$ \quad $\|D\| = |E_1| + |E_2|$ \quad ($E_i \subseteq D \times D$)
- Query $q(x, y) := \exists z, E_1(x, z) \land E_2(z, y)$

This query cannot be enumerated with constant delay\(^1\)

We need to put restrictions on queries and/or databases.

\(^1\)Unless there is a breakthrough with the boolean matrix multiplication.
Example 2 bis

Input:
- Database $D := \langle \{1, \cdots , n\}; E_1; E_2 \rangle$ \hspace{1cm} $\|D\| = |E_1| + |E_2|$ \hspace{1cm} ($E_i \subseteq D \times D$)
- Query $q(x, y) := \exists z, E_1(x, z) \land E_2(z, y)$

and D is a tree!
Example 2 bis

Input:
- Database \( D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle \) \( \|D\| = |E_1| + |E_2| \) (\( E_i \subseteq D \times D \))
- Query \( q(x, y) := \exists z, E_1(x, z) \land E_2(z, y) \)

and \( D \) is a tree!

Given a node \( x \), every solutions \( y \) must be amongst:

- It’s “grandfather”
- It’s “grandchildren”
- It’s “siblings”

\[\begin{align*}
\text{y0} & \quad \text{z} & \quad \text{y1} \quad \text{y2} & \quad \text{y3} \quad \text{y4}
\end{align*}\]
What kind of restrictions?

<table>
<thead>
<tr>
<th>No restriction on the database part</th>
<th>Highly expressive queries (MSO queries)</th>
<th>FO queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only works for a <strong>strict</strong> subset of ACQ</td>
<td>Only works for trees (graphs with bounded tree width)</td>
<td><strong>This talk!</strong></td>
</tr>
</tbody>
</table>

Bagan, Durand, Grandjean

Courcelle, Bagan, Segoufin, Kazana
Problems
For FO queries over a class $\mathcal{C}$ of databases.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Ideal Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-Checking</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Enumeration</td>
<td>$O(1) \cdot O(n)$</td>
</tr>
<tr>
<td>Evaluation</td>
<td>$O(n+m)$</td>
</tr>
<tr>
<td>Counting</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Testing</td>
<td>$O(1) \cdot O(n)$</td>
</tr>
</tbody>
</table>

Ideal running time
Problems
For FO queries over a class $\mathcal{C}$ of databases.

- **Model-Checking**: Is this true? $O(n)$
- **Enumeration**: Enumerate the solutions $O(1) \cdot O(n)$
- **Evaluation**: Compute the entire set $O(n + m)$
- **Counting**: How many solutions? $O(n)$
- **Testing**: Is this tuple a solution? $O(1) \cdot O(n)$

$AW[*]$ complete problem! (when no restriction)
Classes of graphs

- **Local bounded Tree-width**
  - Courcelle et al. 1990
  - Grohe et al. 2011

- **Bounded Tree-width**

- **Excluding minor**

- **Planar**

- **Bounded Degre**
  - Seese, 1996
  - Dvorak et al. 2010

- **Bounded Expansion**
  - Dvorak et al. 2010
  - Segoufin, V. 2017

- **Nowhere Dense**
  - Grohe et al. 2014

- **Local bounded Expansion**
  - Dvorak et al. 2010

- **Classes of graphs**

- **Model-Checking results**

- **Enumeration results**
Classes of graphs

- Somewhere-Dense
  - Dawar, Kreutzer 2009

- Nowhere Dense
  - Grohe et al. 2014

- Local bounded Expansion
  - Dvorak et al. 2010

- Excludeminor
  - Courcelle et al. 1990

- Bounded Tree-width
  - Courcelle et al. 1990

- Bounded Degre
  - Seese, 1996

- Bounded Expansion
  - Dvorak et al. 2010

- BoundedExpansion
  - Dvorak et al. 2010

- Planar

- DENSITY

Model-Checking results

For classes of graphs closed under subgraphs!
Classes of graphs

- Somewhere-Dense
  - Dawar, Kreutzer 2009

- Nowhere Dense
  - Grohe et al. 2014

- Local bounded Expansion
  - Dvorak et al. 2010
  - Segoufin, V. 2017

- Local bounded Tree-width
  - Grohe et al. 2011

- Bounded Expansion
  - Dvorak et al. 2010
  - Segoufin, Kazana 2013

- Bounded Degre
  - Seese, 1996
  - Durand, Grandjean 2007
  - Segoufin, Kazana 2011

- Excludefor minor

- Bounded Tree-width
  - Courcelle et al. 1990
  - Segoufin, Kazana 2013
  - Bagan 2006

- Planar

- DENSITY

- Excludeminor

- Somewhere-Dense
  - Dawar, Kreutzer 2009

- Nowhere Dense
  - Grohe et al. 2014

- Local bounded Expansion
  - Dvorak et al. 2010
  - Segoufin, V. 2017

- Local bounded Tree-width
  - Grohe et al. 2011

- Bounded Expansion
  - Dvorak et al. 2010
  - Segoufin, Kazana 2013

- Bounded Degre
  - Seese, 1996
  - Durand, Grandjean 2007
  - Segoufin, Kazana 2011

- Excludefor minor

- Bounded Tree-width
  - Courcelle et al. 1990
  - Segoufin, Kazana 2013
  - Bagan 2006

- Planar

Model-Checking results
Enumeration results
Classes of graphs

Model-Checking results

Somewhere-Dense
Dawar, Kreutzer 2009

Nowhere Dense
Grohe et al. 2014

limit of tractability

Local bounded Expansion
Dvorak et al. 2010
Segoufin, V. 2017

Bounded Expansion
Dvorak et al. 2010
Segoufin, Kazana 2013

Bounded Degre
Seese, 1996
Durand, Grandjean 2007
Segoufin, Kazana 2011

Bounded Tree-width
Courcelle et al. 1990
Segoufin, Kazana 2013
Bagan 2006

Excluded minor

Excluded minor

Planar

Bagan 2006

DENSITY

This talk

PODS, June 11, 2018 10 / 18
Nowhere dense graphs

Defined by Nešetřil and Ossona de Mendez.¹

Examples:
- graphs with bounded degree
- graphs with bounded tree-width
- planar graphs
- graphs that exclude a minor

Can be defined using:
- the notion of locally excluding a minor
- a small asymptotic ratio edge/vertices
- an ordering of vertices with good properties
- a winning strategy for some two players game

¹First order properties on nowhere dense structures ’10
Results

Theorem: Schweikardt, Segoufin, V.
Over nowhere dense classes of graphs, for every FO query, after a pseudo-linear preprocessing, we can:

- enumerate every solution with constant delay.
- test whether a given tuple is a solution in constant time.

Theorem: Grohe, Schweikardt  *(tomorrow afternoon)*
Over nowhere dense classes of graphs, for every FO query, the number of solution can be computed in pseudo-linear time
Definition

An algorithm is pseudo linear if:

\[ \forall \epsilon > 0, \ \exists N_\epsilon : \begin{cases} \|G\| \leq N_\epsilon \implies \text{Brut force: } O(1) \\ \|G\| > N_\epsilon \implies O(\|G\|^{1+\epsilon}) \end{cases} \]

Examples: \(O(n), \ O(n \log(n)), \ O(n \log^i(n))\)

Counter examples: \(O(n^{1.0001}), \ O(n \sqrt{n})\)
Tools

We use:

- A new Hanf normal form for FO queries.\textsuperscript{1}
- The algorithm for the model checking.\textsuperscript{2}
- Neighbourhood cover.\textsuperscript{2}
- Game characterization of Nowhere-Dense classes.\textsuperscript{2}
- Short-cut pointers dedicated to the enumeration.\textsuperscript{3}

\textsuperscript{1}Grohe, Schweikardt  PODS ’18
\textsuperscript{2}Grohe, Kreutzer, Siebertz  STOC ’14
\textsuperscript{3}Segoufin, V.  ICDT ’17
A neighborhood cover is a set of “representative” neighborhoods.

\( \mathcal{X} := X_1, \ldots, X_n \) is a \( r \)-neighborhood cover if it has the following properties:

- \( \forall a \in G, \ \exists X \in \mathcal{X}, \ N_r(a) \subseteq X \)
- \( \forall X \in \mathcal{X}, \ \exists a \in G, \ X \subseteq N_{2r}(a) \)
- \( \forall a \in G, \ |\{ i | a \in X_i \}| \) is pseudo-constant (smaller than \( |G|^{\epsilon} \))
The game characterization

**Definition : \((\ell, r)\)-Splitter game**

A graph \(G\) and two players, Splitter and Connector. Each turn:

- Connector picks a node \(c\)
- Splitter picks a node \(s\)
- \(G' = N_r^G(c) \setminus s\)

If in less than \(\ell\) rounds the graph is empty, Splitter wins.
The game characterization

**Definition :** \((\ell, r)\)-Splitter game

A graph \(G\) and two players, Splitter and Connector. Each turn:
- Connector picks a node \(c\)
- Splitter picks a node \(s\)
- \(G' = N_r^G(c) \setminus s\)

If in less than \(\ell\) rounds the graph is empty, Splitter wins.

**Theorem**

\(\mathcal{C}\) is nowhere dense if and only if there is a function \(f_\mathcal{C}\) such that for every \(G \in \mathcal{C}\) and every \(r \in \mathbb{N}\):

Splitter has a winning strategy for the \((f_\mathcal{C}(r), r)\)-splitter game on \(G\).
How to use the game

Here, the query is \( q(x, y) := \exists z, \ E(x, z) \land E(z, y) \) (distance two query)
How to use the game

Here, the query is \( q(x, y) := \exists z, E(x, z) \land E(z, y) \) (distance two query)

- when there is still a 2-path not using \( s \)
  - the new query is: \( q(x, y) \)

- when \( s \) is on the only short path from \( a \) to \( b \)
  - the new query is: \( R_1(x) \land R_1(y) \lor q(x, y) \)

- when \( a = s \) (similarly for \( b = s \))
  - the new query is: \( R_2(y) \)
Future work

- Classes of graphs that are not closed under subgraphs \(^1\)

- Enumeration with update:
  What happens if a small change occurs after the preprocessing?

  \textit{Existing results for: words,} \(^2\) \textit{graphs with bounded degree} \(^3\) \textit{and ACQ} \(^4\).

---

\(^1\) Gajarský, Kreutzer, Nešetřil, Ossona de Mendez, Pilipczuk, Siebertz, Toruńczyk \(\text{ICALP '18}\)

\(^2\) Niewerth, Segoufin \(\text{PODS '18}\) (in two talks!)

\(^3\) Berkholz, Keppeler, Schweikardt \(\text{ICDT '17}\)

\(^4\) Berkholz, Keppeler, Schweikardt \(\text{PODS '17 & ICDT '18}\)
Future work

- Classes of graphs that are not closed under subgraphs \(^1\)
- Enumeration with update:
  What happens if a small change occurs after the preprocessing?

Existing results for: words,\(^2\) graphs with bounded degree,\(^3\) and ACQ.\(^4\)

Thank you!

Questions?

---

\(^1\) Gajarský, Kreutzer, Nešetřil, Ossona de Mendez, Pilipczuk, Siebertz, Toruńczyk ICALP ’18
\(^2\) Niewerth, Segoufin PODS ’18 (in two talks!)
\(^3\) Berkholz, Keppeler, Schweikardt ICDT ’17
\(^4\) Berkholz, Keppeler, Schweikardt PODS ’17 & ICDT ’18