Constant delay enumeration for FO queries and nowhere dense graphs

Alexandre Vigny\textsuperscript{1}

Join work with: Nicole Schweikardt\textsuperscript{2} and Luc Segoufin\textsuperscript{3}

\textsuperscript{1}Université Paris Diderot, Paris
\textsuperscript{2}Humboldt-Universität zu Berlin
\textsuperscript{3}ENS Ulm, Paris

June 1, 2018
Modelization

- 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€?

- For practical reasons:
  \[ 50^{10} \text{ solutions is not easy to store / display!} \]

- 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 \( D \rightarrow \text{Index} \ I \)

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

- **STEP 2: Enumeration**

  Enumerate the solutions: \( \text{Index} \ I \rightarrow x_1, x_2, x_3, x_4, \cdots \)

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

  Constant delay enumeration after linear preprocessing
Example 1

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

```
D
(1,1)
(1,2)
(1,6)
  
  ...
(2,3)
  
  ...
(i,j)
(i,j+1)
(i,j+3)
  
  ...
(n,n)
```
Example 1

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

<table>
<thead>
<tr>
<th>D</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(1,2)</td>
<td>(1,2)</td>
</tr>
<tr>
<td>(1,6)</td>
<td>(1,6)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,3)</td>
<td>(2,3)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(i,j)</td>
<td>(i,j)</td>
</tr>
<tr>
<td>(i,j+1)</td>
<td>(i,j+1)</td>
</tr>
<tr>
<td>(i,j+3)</td>
<td>(i,j+3)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(n,n)</td>
<td>(n,n)</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
</tr>
</tbody>
</table>
Example 1

Input:
- Database $D := \langle\{1, \ldots, n\}; E\rangle$ \hspace{1cm} $\|D\| = |E|$ \hspace{1cm} ($E \subseteq D \times D$)
- 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>(2,4)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(i,j)</td>
<td>(i,j)</td>
<td>(1,6)</td>
</tr>
<tr>
<td>(i,j+1)</td>
<td>(i,j+1)</td>
<td>(2,4)</td>
</tr>
<tr>
<td>(i,j+3)</td>
<td>(i,j+3)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>...</td>
<td>...</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

June 1, 2018
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 $q(x, y) := \exists z, \ E_1(x, z) \land E_2(z, y)$

Compute the set of solutions

\begin{equation}
A : \text{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}
\end{equation}

\begin{equation}
B : \text{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}
\end{equation}

\begin{equation}
C : \text{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}
\end{equation}

Boolean matrix multiplication
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 $q(x, y) := \exists z, E_1(x, z) \land E_2(z, y)$

$B :$ Adjacency matrix of $E_2$

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

$A :$ Adjacency matrix of $E_1$

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

$C :$ Result matrix

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

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

This query cannot be enumerated with constant delay$^1$

---

$^1$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.
Which restrictions?

<table>
<thead>
<tr>
<th>No restriction on the database part</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
</tr>
<tr>
<td>Only works for queries are conjunctive, acyclic and free-connex</td>
</tr>
<tr>
<td>Bagan, Durand, Grandjean</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Highly expressive queries (MSO queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
</tr>
<tr>
<td>Only works for trees (Graphs with bounded tree width)</td>
</tr>
<tr>
<td>Courcelle, Bagan, Segoufin, Kazana</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FO queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
</tr>
<tr>
<td>This talk!</td>
</tr>
</tbody>
</table>

Alexandre Vigny

Enumeration & nowhere dense graphs

June 1, 2018
Other problems

For FO queries over a class $\mathcal{C}$ of databases.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-Checking</td>
<td>Is this true?</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Enumeration</td>
<td>Enumerate the solutions</td>
<td>$O(1) \cdot O(n)$</td>
</tr>
<tr>
<td>Counting</td>
<td>How many solutions?</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Testing</td>
<td>Is this tuple a solution?</td>
<td>$O(1) \cdot O(n)$</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Compute the entire set</td>
<td>$O(n + m)$</td>
</tr>
</tbody>
</table>

\[ AW_{\ast} \]\[ complete \] problem!

Alexandre Vigny  
Enumeration & nowhere dense graphs  
June 1, 2018  8 / 22
Other problems

For FO queries over a class $\mathcal{C}$ of databases.

Model-Checking : Is this true ? $O(n)$

Enumeration : Enumerate the solutions $O(1) \circ O(n)$

Counting : How many solutions ? $O(n)$

Testing : Is this tuple a solution ? $O(1) \circ O(n)$

Evaluation : Compute the entire set $O(n + m)$

AW[$\ast$] complete problem!
Classes of graphs closed under taking sub-graphs

- **Bounded Degre**
  - Seese, 1996

- **Bounded Tree-width**
  - Courcelle et al., 1990
  - Grohe et al., 2011
  - Durand, Grandjean, 2007
  - Segoufin, Kazana, 2013

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

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

- **Excluded minor**

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

- **Planar**

- **Bounded Degre**
  - Seese, 1996

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

Model-Checking results
Classes of graphs closed under taking sub-graphs

- Somewhere-Dense
  - Dawar, Kreutzer 2009

- Nowhere Dense
  - Grohe et al. 2014

- Local bounded Expansion
  - Dvorak et al. 2010

- Bounded Expansion
  - Dvorak et al. 2010

- Bounded Degre
  - Seese. 1996

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

- Excludeminor

- Bounded Tree-width
  - Courcelle et al. 1990

- Planar

- DENSITY

- limit of tractability

Model-Checking results
Classes of graphs closed under taking sub-graphs

Model-Checking results

Enumeration results

Somewhere-Dense
Dawar, Kreutzer 2009

Nowhere Dense
Grohe et al. 2014

Local bounded Expansion
Dvorak et al. 2010
Segoufin, Kazana 2013

Bounded Expansion
Dvorak et al. 2010
Segoufin, Kazana 2013

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

Local bounded Tree-width
Grohe et al. 2011

Exclude minor

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

Planar

DENSITY

limit of tractability
Classes of graphs closed under taking sub-graphs

- Somewhere-Dense
  - Dawar, Kreutzer 2009

- Nowhere Dense
  - Grohe et al. 2014

- 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

- Excludeminor
  - Grohe et al. 2011

- Planar

Model-Checking results
Enumeration results

DENSITY

With: Nicole Schweikardt
Luc Segoufin

PODS '18
Our results

Theorem (Schweikardt, Segoufin, V. 18’)

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

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

**Definition**

A function $f$ is pseudo linear if and only if:

$$
\forall \epsilon > 0, \ \exists N_\epsilon \in \mathbb{N}, \ \forall n \in \mathbb{N}, \ n > N_\epsilon \implies f(n) \leq n^{1+\epsilon} 
$$

$$
n \ll n \log^i(n) \ll \text{pseudo-linear} \ll n^{1.0001} \ll n\sqrt{n}
$$

“Pseudo-linear $\approx n \log^i(n)$”

“Pseudo-constant $\approx \log^i(n)$”
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 play the \((\ell, r)\)-Splitter game on a graph \(G\) ?

- If \(G\) is a star, Splitter wins in 2 rounds.
- If \(G\) is a path, Splitter wins in \(\log(r)\) rounds.
- If \(G\) is a tree, Splitter wins in \(r\) rounds.
- If \(G\) has degree \(d\), splitter wins in \(d^r\) rounds.
- If \(G\) is a clique of size \(> \ell\), Splitter looses the \((\ell, r)\)-splitter game.
Neighborhood cover

A neighborhood cover is a set of “representative” neighborhoods.

\( \mathcal{X} := X_1, \ldots, X_n \) is a \((r, 2r)\) neighborhood cover if it has the following properties:

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

- \( q_1(x, y) := \exists z \ E(x, z) \land E(z, y) \)
  
  (The distance two query)

- \( q_2(x, y) := \neg q_1(x, y) \)
  
  (Nodes that are far apart)
How to use the game 1/2

$G$ is now fixed

Goal: Given a node $a$ we want to enumerate all $b$ such that $q_1(a, b)$. (Here $r = 2$)

- Base case: If Splitter wins the $(1, r)$-Splitter game on $G$. Then $G$ is edgeless and there is no solution!

- By induction: assume that there is an algorithm for every $G'$ such that Splitter wins the $(\ell, r)$-Splitter game on $G'$. 
How to use the game 2/2

Here, Splitter wins the \((\ell + 1, r)\)-game on \(G\).

Idea :

1. Compute some new graphs on which Splitter wins the \((\ell, r)\) game.
2. Apply the induce algorithm for a particular query.
3. Enumerate those solutions.
How to use the game 2/2

Here, Splitter wins the \((\ell + 1, r)\)-game on \(G\).

Idea:
1. Compute some new graphs on which Splitter wins the \((\ell, r)\) game.
2. Apply the induce algorithm for a particular query.
3. Enumerate those solutions.

For every bags \(X\) of the \((2,4)\)-neighborhood cover, \(X' := X \setminus \{s\}\).

For every \((a, b) \in G^2\) we have:

\[
G \models q_1(a, b) \iff \bigvee_{X \in \mathcal{X}} X \models q_1(a, b) \iff \mathcal{X}(a) \models q_1(a, b)
\]
How to use the game 2/2

Here, Splitter wins the \((\ell + 1, r)\)-game on \(G\).

Idea :

1. Compute some new graphs on which Splitter wins the \((\ell, r)\) game.
2. Apply the induce algorithm for a particular query.
3. Enumerate those solutions.

For every bags \(X\) of the \((2,4)\)-neighborhood cover, \(X' := X \setminus \{s\}\).

For every \((a, b) \in G^2\) we have:

\[
G \models q_1(a, b) \iff \bigvee_{X \in \mathcal{X}} X \models q_1(a, b) \iff \mathcal{K}(a) \models q_1(a, b)
\]

The new graph is \(\mathcal{K}(a)\)

Then, Splitter delete a node!
The new queries

when there is still a 2-path not using $s$

the new query is:

$q_1(x, y)$

when $s$ is on the only short path from $a$ to $b$

the new query is:

$R_1(x) \land R_2(y)$

when $a = s$

(similarly for $b = s$)

the new query is:

$R_2(y)$
The second query

\[ q_2(x, y) := \text{dist}(x, y) > 2 \]

Two kinds of solutions:

- \( b \in X(a) \) (similar to the previous example)
- \( b \not\in X(a) \) We need something else!
The second query

\[ q_2(x, y) := \text{dist}(x, y) > 2 \]

Two kinds of solutions:

- \( b \in \mathcal{X}(a) \) (similar to the previous example)
- \( b \not\in \mathcal{X}(a) \) We need something else!

Goal: given a bag \( X \), enumerate all \( b \not\in X \)
The shortcut pointers

Given $X$ we want to enumerate all $b$ such that $b \not\in X$. 
The shortcut pointers

Given $X$ we want to enumerate all $b$ such that $b \notin X$.

$$NEXT(b, X) := \min \{ b' \in G \mid b' \geq b \land b' \notin X \}$$
The shortcut pointers

Given $X$ we want to enumerate all $b$ such that $b \not\in X$.

$$NEXT(b, X) := \min \{ b' \in G \mid b' \geq b \land b' \not\in X \}$$

For all $X \in \mathcal{X}$ with $b_{\text{max}} \in X$, we have $NEXT(b_{\text{max}}, X) = \text{NULL}$
The shortcut pointers

Given $X$ we want to enumerate all $b$ such that $b \not\in X$.

$$NEXT(b, X) := \min\{b' \in G \mid b' \geq b \land b' \not\in X \}$$

For all $X \in \mathcal{X}$ with $b_{\text{max}} \in X$, we have $NEXT(b_{\text{max}}, X) = \text{NULL}$

$$NEXT(b, X) \in \{b + 1, \, NEXT(b + 1, X)\}$$
The shortcut pointers

Given $X$ we want to enumerate all $b$ such that $b \notin X$.

\[
NEXT(b, X) := \min_{b \in X} \{b' \in G \mid b' \geq b \land b' \notin X\}
\]

For all $X \in \mathcal{X}$ with $b_{\text{max}} \in X$, we have $NEXT(b_{\text{max}}, X) = \text{NULL}$

\[
NEXT(b, X) \in \{b+1, NEXT(b+1, X)\}
\]
Recap

We use:

- A new Hanf normal form for FO queries.\(^1\)
- The algorithm for the model checking.\(^2\)
- Neighbourhood cover.\(^2\)
- Game characterization of nowhere dense classes.\(^2\)
- Short-cut pointers dedicated to the enumeration.\(^3\)

We can:

- Enumerate with constant delay after pseudo-linear preprocessing.
- Test in constant time after pseudo-linear preprocessing.

\(^1\) Grohe, Schweikardt '18
\(^2\) Grohe, Kreutzer, Siebertz '14
\(^3\) Segoufin, V. '17
Future work

- Classes of graphs that are not closed under subgraphs

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

  *Existing results for: words, graphs with bounded tree-width or bounded degree.*
Future work

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

  *Existing results for: words, graphs with bounded tree-width or bounded degree.*

Thank you!

Questions?