Constant delay enumeration for FO queries over nowhere dense graphs

Nicole Schweikardt[1], Luc Segoufin[2] and **Alexandre Vigny**[3]

[1]Humboldt-Universität zu Berlin, Berlin

[2]Inria, ENS Ulm, Paris

[3]Université Paris Diderot, Paris

PODS, June 11, 2018

Query evaluation

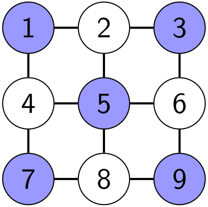
- Query q *small*
- Database D *huge*
- Compute $q(D)$ *gigantic*

Examples :

query *q*
first order logic

$$q(x, y) := \exists z(B(x) \wedge E(x, z) \wedge \neg E(y, z))$$

database *D*
relational structure



solutions *q(D)*
set of tuples

$\{(1,2)\ (1,3)\ (1,4)$
$(1,6)\ (1,7)\ \cdots$
$(3,1)\ (3,2)\ (3,4)$
$(3,6)\ (3,7)\ \cdots$
$\cdots\}$

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

Goal : output solutions one by one (no repetition)

- STEP 1: Preprocessing

Prepare the enumeration : Database $D \longrightarrow$ Index I

Preprocessing time : $f(k) \cdot n \rightsquigarrow O(n)$

- STEP 2 : Enumeration

Enumerate the solutions : Index $I \longrightarrow \overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_4}, \cdots$

Delay : $O(f(k)) \rightsquigarrow O(1)$

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

Example 1

Input :

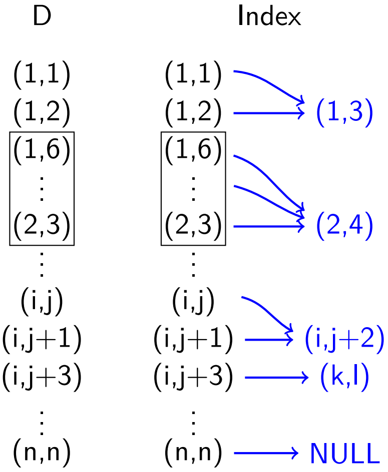
- Database $D := \langle\{1,\cdots,n\};E\rangle$ $\|D\| = |E|$
- Query $q(x,y) := \neg E(x,y)$

$$
\begin{array}{c}
\text{D} \\
\\
(1,1) \\
(1,2) \\
\boxed{\begin{array}{c}(1,6)\\ \vdots \\ (2,3)\end{array}} \\
\vdots \\
(i,j) \\
(i,j+1) \\
(i,j+3) \\
\vdots \\
(n,n)
\end{array}
$$

Example 1

Input :

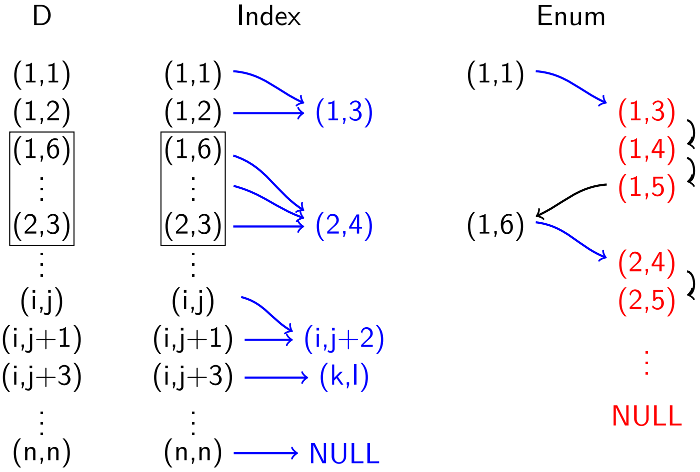
- Database $D := \langle \{1, \cdots, n\}; E \rangle$ $\qquad \|D\| = |E|$
- Query $q(x,y) := \neg E(x,y)$



Example 1

Input :

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



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 $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

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 $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

B : Adjacency matrix of E_2

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

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

A : Adjacency matrix of E_1

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

C : 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 $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

B : Adjacency matrix of E_2

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

$$\begin{pmatrix} E_1(1,1) & \dots & E_1(1,i) & \dots & E_1(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(x,1) & \dots & E_1(x,z) & \dots & E_1(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(n,1) & \dots & E_1(n,z) & \dots & E_1(n,n) \end{pmatrix} \quad \begin{pmatrix} q(1,1) & \dots & q(1,y) & \dots & q(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(x,1) & \dots & q(x,y) & \dots & q(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(n,1) & \dots & q(n,y) & \dots & q(n,n) \end{pmatrix}$$

A : Adjacency matrix of E_1

C : Result matrix

Compute the set of solutions

=

boolean matrix multiplication

Example 2

Input :

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

B : Adjacency matrix of E_2

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

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

A : Adjacency matrix of E_1

C : Result matrix

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 $q(x, y) := \exists z,\ E_1(x, z) \wedge E_2(z, y)$

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 $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

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

and D is a tree!

Example 2 bis

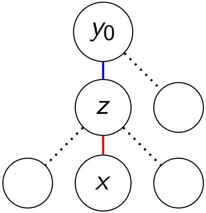
Input :

- 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 $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

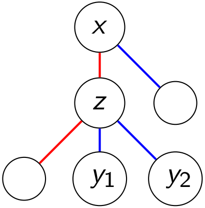
and D is a tree!

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

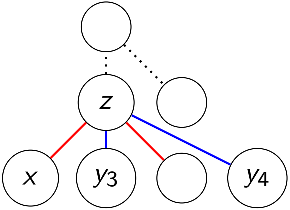
It's "grandfather"



It's "grandchildren"



It's "siblings"



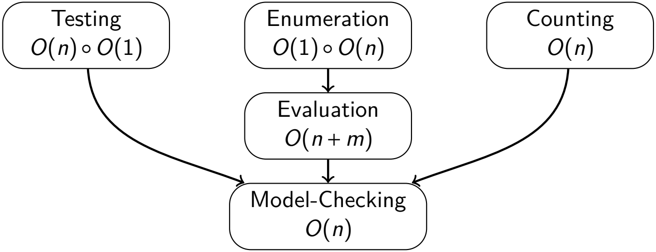
What kind of restrictions?

No restriction on the database part	Highly expressive queries (MSO queries)	FO queries
$\Downarrow$	$\Downarrow$	$\Downarrow$
Only works for a **strict** subset of ACQ	Only works for trees (graphs with bounded tree width)	**This talk !**
Bagan, Durand, Grandjean	Courcelle, Bagan, Segoufin, Kazana	

Problems

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

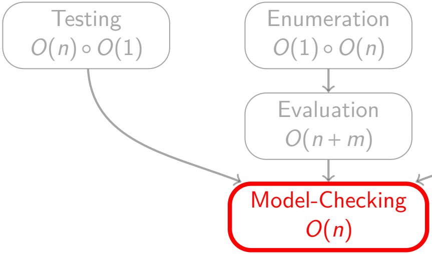
			Ideal running time
Model-Checking	:	Is this true ?	$O(n)$
Enumeration	:	Enumerate the solutions	$O(1) \circ O(n)$
Evaluation	:	Compute the entire set	$O(n+m)$
Counting	:	How many solutions ?	$O(n)$
Testing	:	Is this tuple a solution ?	$O(1) \circ O(n)$



Problems

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

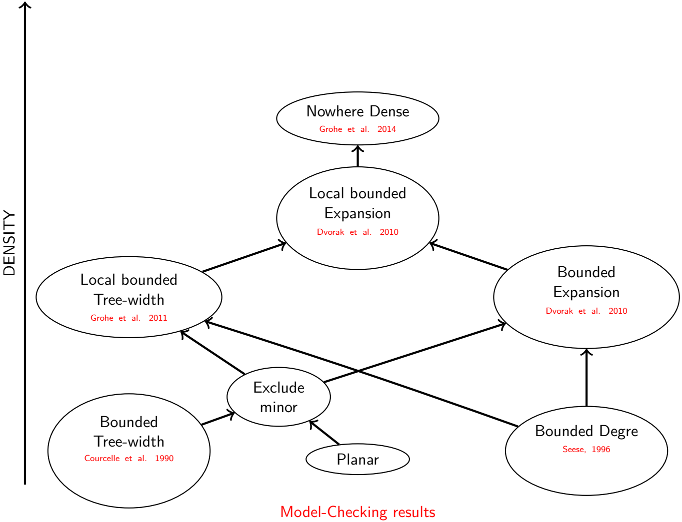
			Ideal running time
Model-Checking	:	Is this true ?	$O(n)$
Enumeration	:	Enumerate the solutions	$O(1) \circ O(n)$
Evaluation	:	Compute the entire set	$O(n+m)$
Counting	:	How many solutions ?	$O(n)$
Testing	:	Is this tuple a solution ?	$O(1) \circ O(n)$



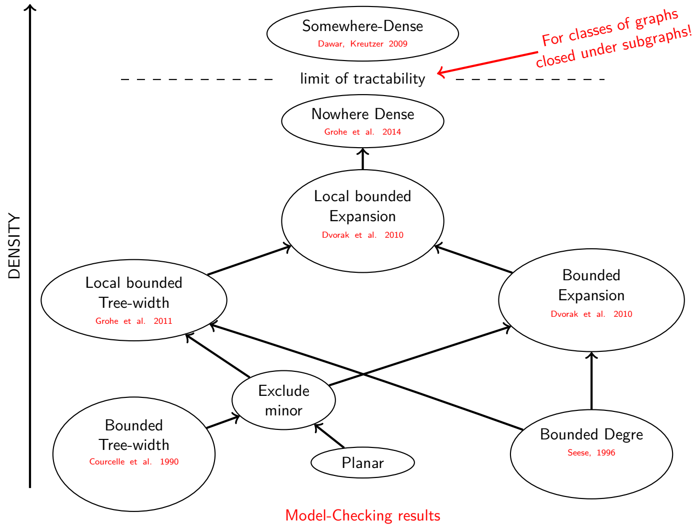
AW[$*$] complete problem!

(when no restriction)

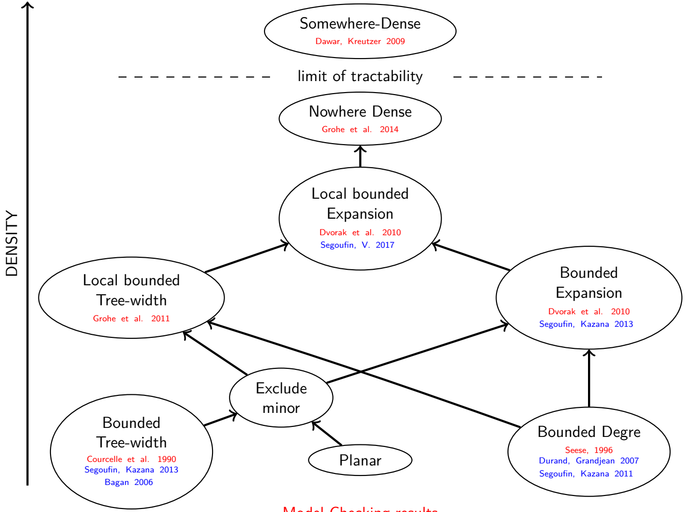
Classes of graphs



Classes of graphs

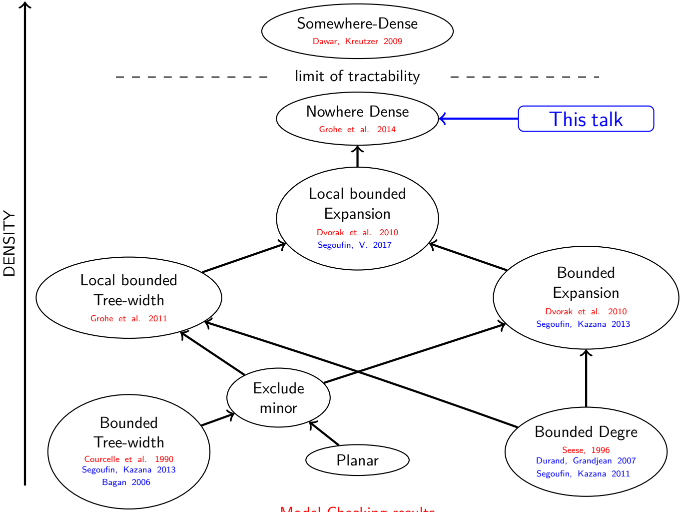


Classes of graphs



Model-Checking results
Enumeration results

Classes of graphs



Model-Checking results
Enumeration results

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

[1] First 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

Pseudo-linear ?

Definition

An algorithm is pseudo linear if:

$$\forall \epsilon > 0, \quad \exists N_\epsilon : \qquad \begin{array}{l} \bullet \|G\| \leq N_\epsilon \implies \text{Brut force: } O(1) \\ \bullet \|G\| > N_\epsilon \implies O(\|G\|^{1+\epsilon}) \end{array}$$

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.[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]

[1] Grohe, Schweikardt PODS '18
[2] Grohe, Kreutzer, Siebertz STOC '14
[3] Segoufin, V. ICDT '17

Neighborhood cover

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

$\mathscr{X} := X_1, \dots, X_n$ is a r-neighborhood cover if it has the following properties:

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

The game characterization

Definition : (ℓ, 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 ℓ rounds the graph is empty, Splitter wins.

The game characterization

Definition : (ℓ, 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 ℓ rounds the graph is empty, Splitter wins.

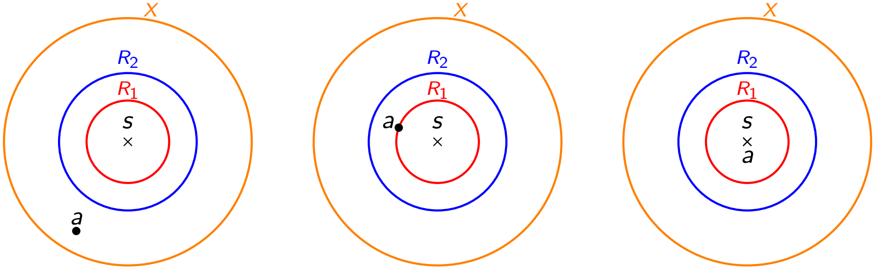
Theorem

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

Splitter has a wining strategy for the $(f_{\mathscr{C}}(r), r)$-splitter game on G.

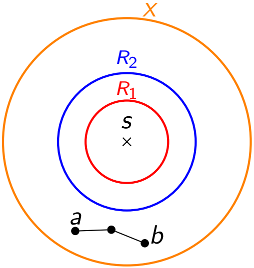
How to use the game

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



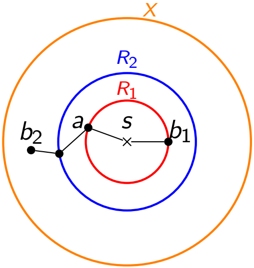
How to use the game

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



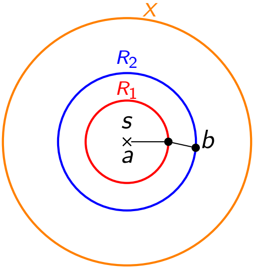
when there is still a 2-path not using s

the new query is:
$q(x,y)$



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

the new query is:
$R_1(x) \wedge R_1(y)$
$\vee q(x,y)$



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 ?

  Existing results for: words,[2] *graphs with bounded degree* [3] *and* ACQ [4]*.*

[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

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