Constant delay enumeration for FO queries and nowhere dense graphs

PART 1

Alexandre Vigny[1]

Join work with: Nicole Schweikardt[2] and Luc Segoufin[3]

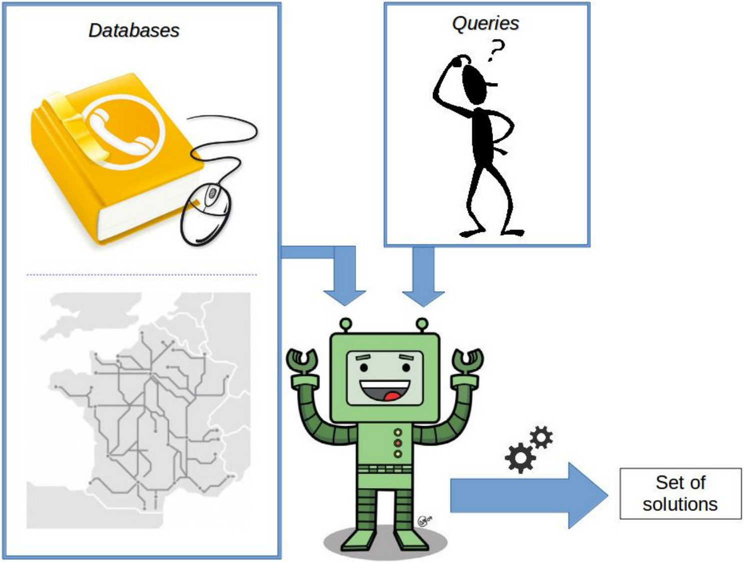
[1]Université Paris Diderot, Paris

[2]Humboldt-Universität zu Berlin

[3]ENS Ulm, Paris

March 15, 2018

Introduction



Modelization

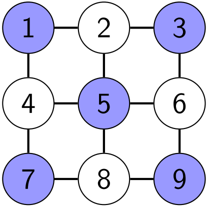
- Query q *small*
- Database D *huge*
- Compute $q(D)$ *gigantic*

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€ ?

- For practical reasons:

  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)

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

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

Example 1

Input :

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

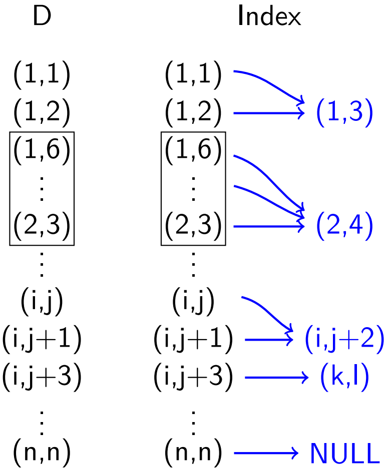
D

(1,1)
(1,2)
(1,6)
⋮
(2,3)
⋮
(i,j)
(i,j+1)
(i,j+3)
⋮
(n,n)

Example 1

Input :

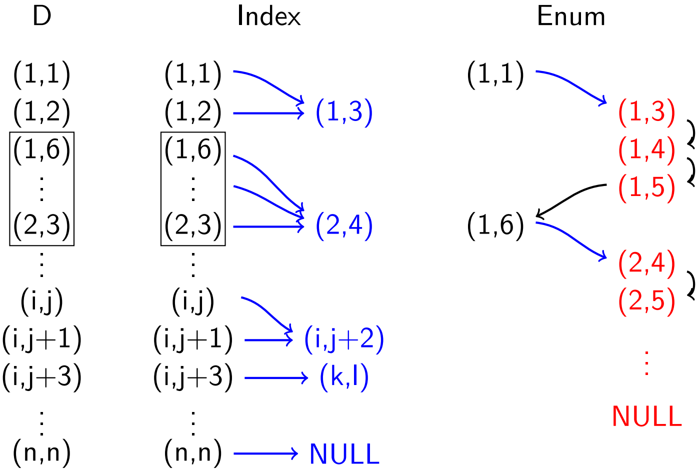
- Database $D := \langle\{1,\cdots,n\};E\rangle$ $\quad \|D\| = |E| \quad (E \subseteq D \times D)$
- Query $q(x,y) := \neg E(x,y)$



Example 1

Input :

- Database $D := \langle\{1,\cdots,n\};E\rangle$ $\|D\| = |E|$ $(E \subseteq D \times D)$
- 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

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

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

A : Adjacency matrix of E_1

C : Result matrix

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

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

Compute the set of solutions

=

boolean matrix multiplication

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

A : Adjacency matrix of E_1

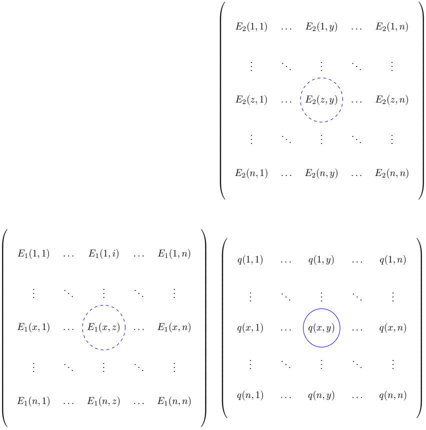
C : Result matrix

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



A : Adjacency matrix of E_1

C : Result matrix

- 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

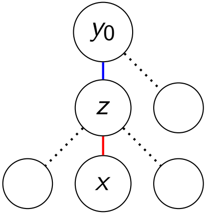
Input :

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

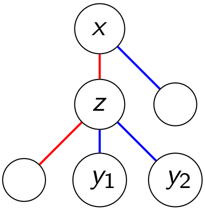
And D is a tree !

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

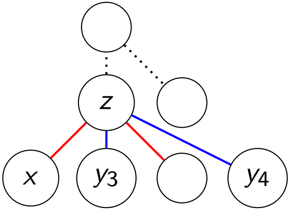
It's “grandfather”



It's “grandsons”



It's “siblings”



Which restrictions ?

No restriction on the database part	Highly expressive queries (MSO queries)	Little bit of both

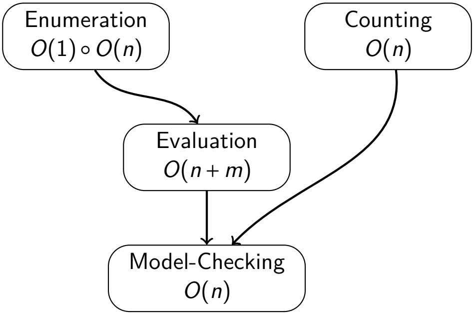
Which restrictions ?

No restriction on the database part	Highly expressive queries (MSO queries)	Little bit of both
⇓	⇓	⇓
Only works for queries are conjunctive, acyclic and free-connex	Only works for trees (Graphs with bounded tree width)	**This talk !**
Bagan, Durand, Grandjean	Courcelle, Bagan, Segoufin, Kazana	(answer in two slides !)

Other problems

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

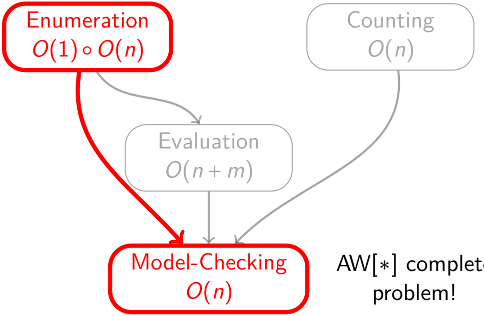
Model-Checking	:	Is this true ?	$O(n)$
Enumeration	:	Enumerate the solutions	$O(1) \circ O(n)$
Counting	:	How many solutions ?	$O(n)$
Evaluation	:	Compute the entire set	$O(n+m)$



Other problems

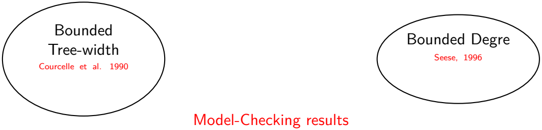
For FO queries over a class $\mathscr{C}$ of databases.

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

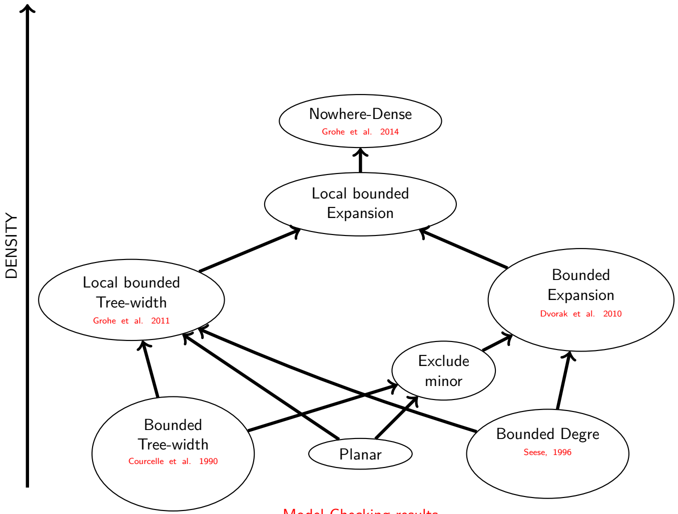


AW[∗] complete problem!

Classes of graphs closed under taking sub-graphs

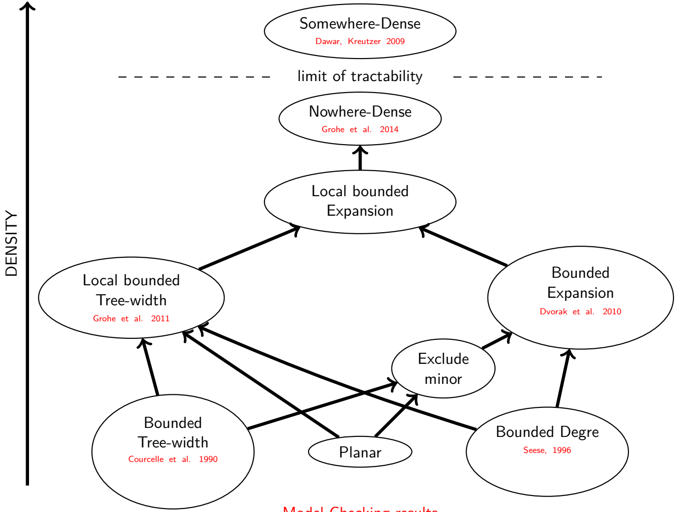


Classes of graphs closed under taking sub-graphs



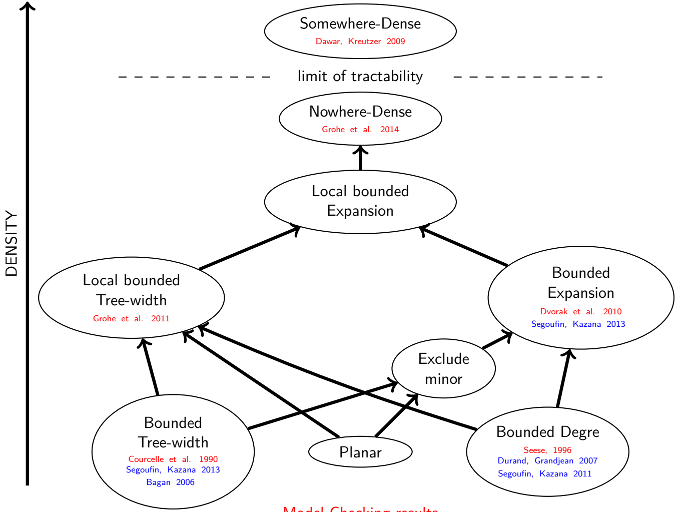
Model-Checking results

Classes of graphs closed under taking sub-graphs



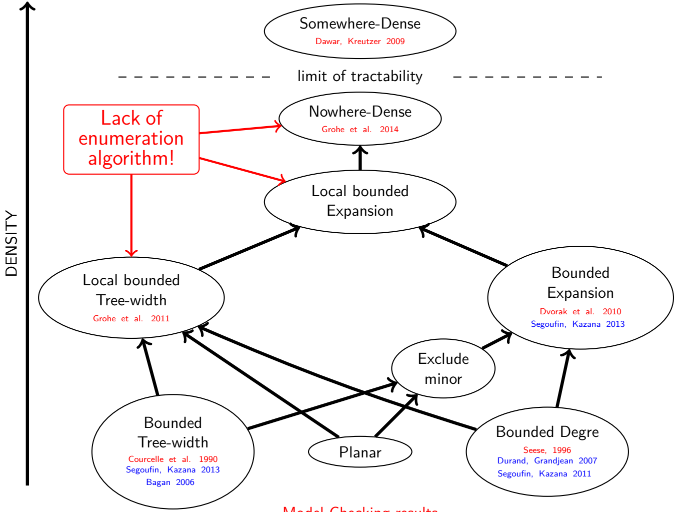
Model-Checking results

Classes of graphs closed under taking sub-graphs



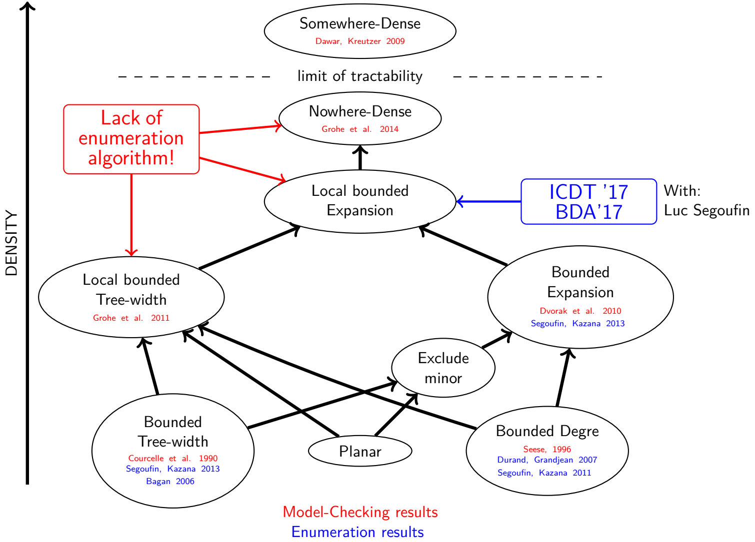
Model-Checking results
Enumeration results

Classes of graphs closed under taking sub-graphs

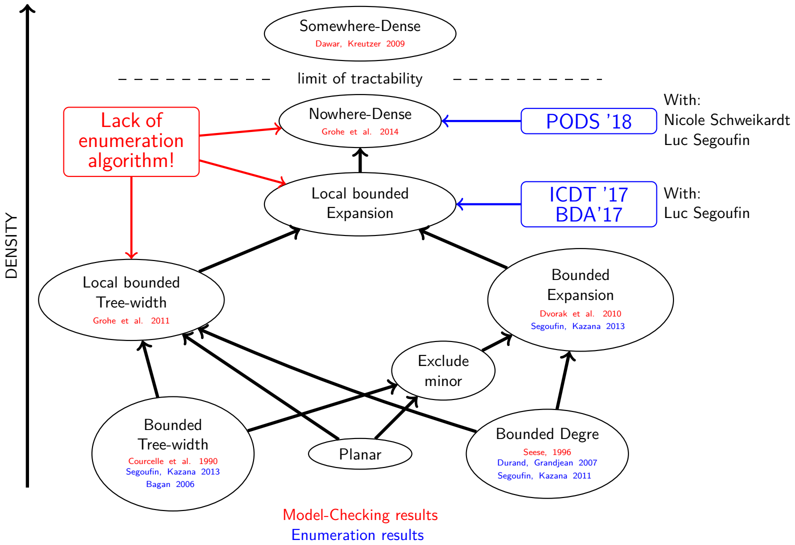


Model-Checking results
Enumeration results

Classes of graphs closed under taking sub-graphs



Classes of graphs closed under taking sub-graphs



Our results

Theorem (Segoufin, V. 17')

Over classes of graphs with *local bounded expansion*, for every FO query, after a pseudo-linear preprocessing, we can:

- enumerate with constant delay every solutions.
- test in constant time whether a given tuple is a solution.
- compute in constant time the number of solutions.

Our results

Theorem (Segoufin, V. 17')

Over classes of graphs with *local bounded expansion*, for every FO query, after a pseudo-linear preprocessing, we can:

- enumerate with constant delay every solutions.
- test in constant time whether a given tuple is a solution.
- compute in constant time the number of solutions.

Theorem (Schweikardt, Segoufin, V. 18')

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

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

Pseudo-linear ?

A function f is pseudo linear if and only if:

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

$$n \ll n\log^i(n) \ll \text{pseudo-linear} \ll n^{1,0001} \ll n\sqrt{n}$$

$$\text{“Pseudo-linear} \approx n\log^i(n)\text{”}$$

$$\text{“Pseudo-constant} \approx \log^i(n)\text{”}$$

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 ?