Twin-width

Édouard Bonnet, Colin Geniet, Eun Jung Kim,
Stéphan Thomassé, and Rémi Watrigant

ENS Lyon, LIP

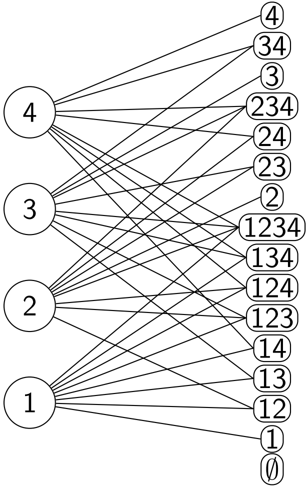
Frontiers of Parameterized Complexity seminar,
October 1st 2020

Cograph generalization attempt

Iteratively identify **near** twins

Cograph generalization attempt

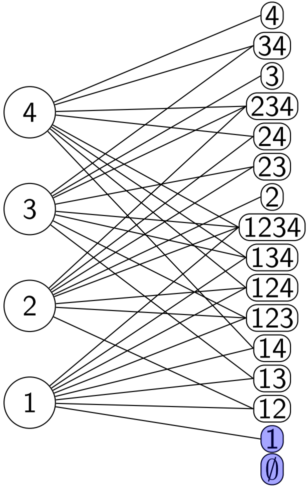
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

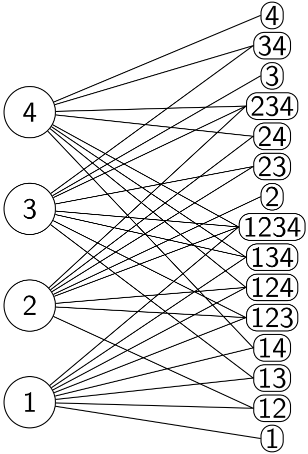
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

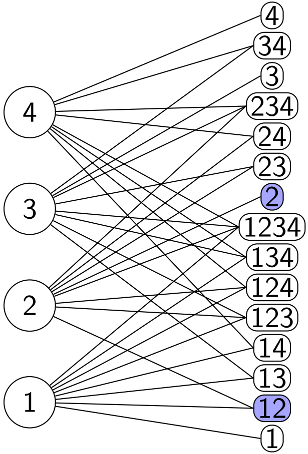
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

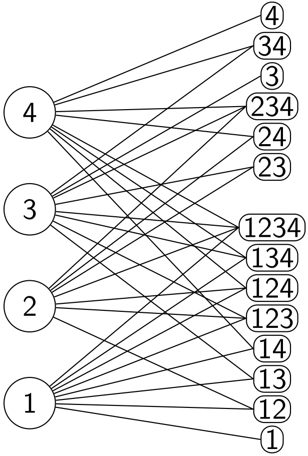
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

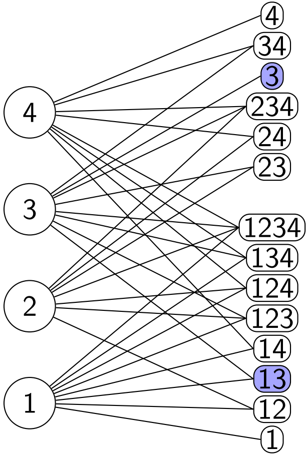
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

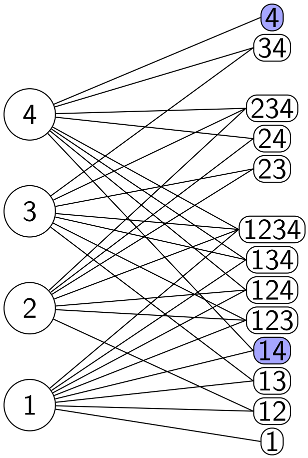
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

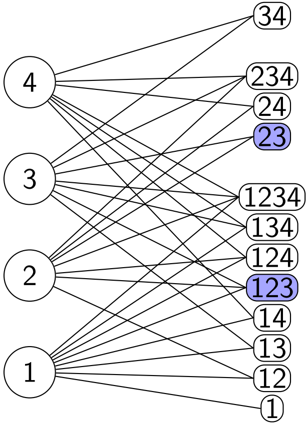
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

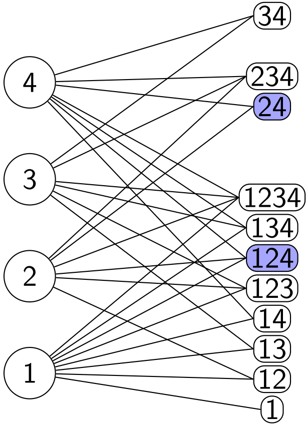
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

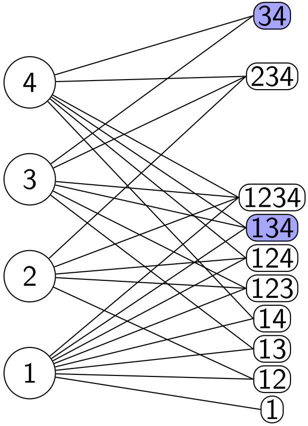
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

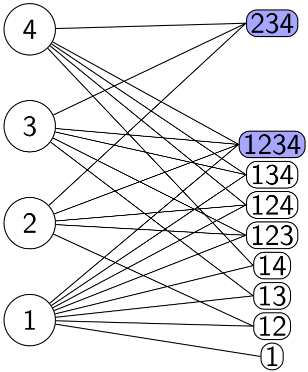
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

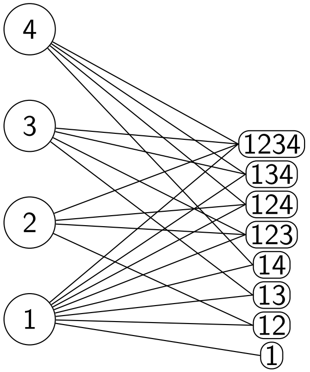
Iteratively identify **near** twins



This complicated graph passes the test

Cograph generalization attempt

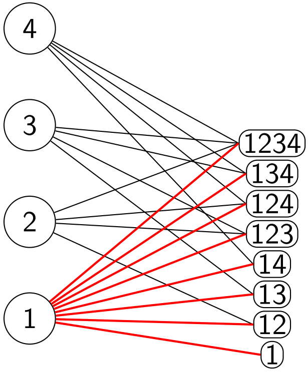
Iteratively identify **near** twins



This complicated graph passes the test

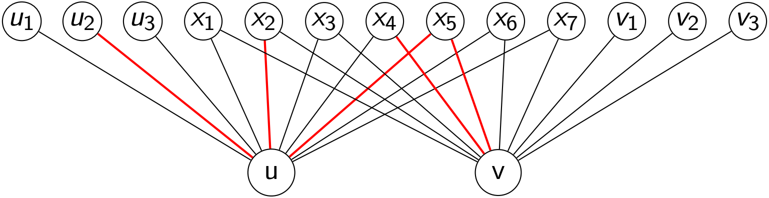
Cograph generalization

Iteratively identify **near** twins and **keep the error degree small**



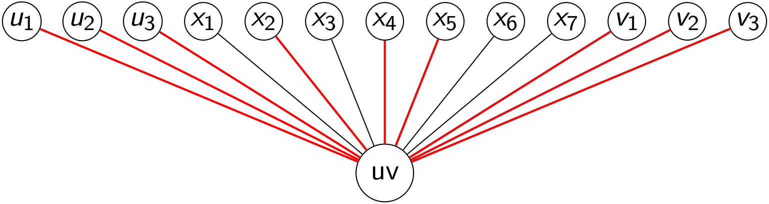
It would not with that further restriction

Contraction and trigraph



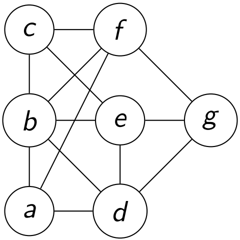
Trigraph: non-edges, edges, and red edges (error)

Contraction and trigraph



edges to $N(u) \triangle N(v)$ turn red, for $N(u) \cap N(v)$ red is absorbing

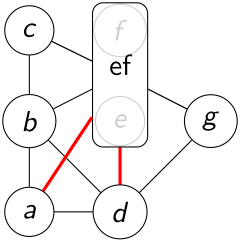
Contraction sequence and twin-width



Maximum red degree = 0

overall maximum red degree = 0

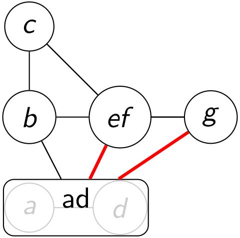
Contraction sequence and twin-width



Maximum red degree = 2

overall maximum red degree = 2

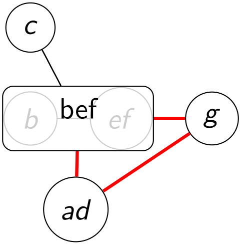
Contraction sequence and twin-width



Maximum red degree = 2

overall maximum red degree = 2

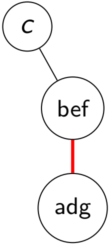
Contraction sequence and twin-width



Maximum red degree = 2

overall maximum red degree = 2

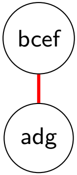
Contraction sequence and twin-width



Maximum red degree = 1

overall maximum red degree = 2

Contraction sequence and twin-width



Maximum red degree = 1

overall maximum red degree = 2

Contraction sequence and twin-width

Maximum red degree $= 0$

overall maximum red degree = 2

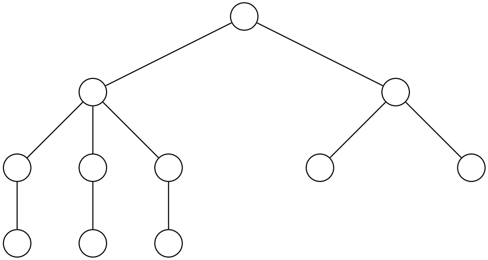
Contraction sequence and twin-width

Sequence of 2-contractions or 2-sequence, twin-width at most 2

Maximum red degree = 0

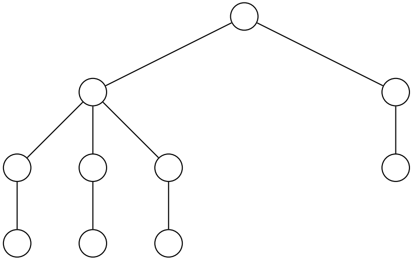
overall maximum red degree = 2

Graphs with bounded twin-width – trees



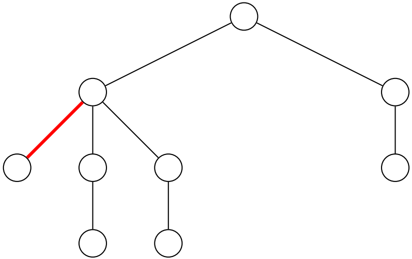
If possible, contract two twin leaves

Graphs with bounded twin-width – trees



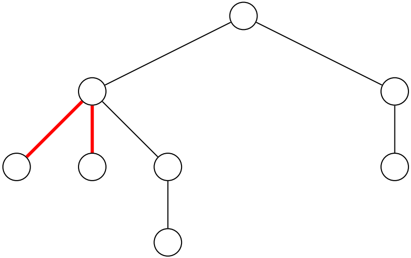
If not, contract a deepest leaf with its parent

Graphs with bounded twin-width – trees



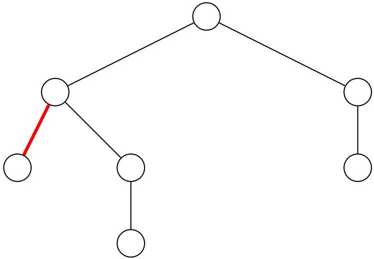
If not, contract a deepest leaf with its parent

Graphs with bounded twin-width – trees



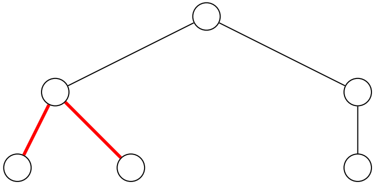
If possible, contract two twin leaves

Graphs with bounded twin-width – trees



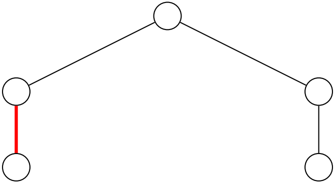
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



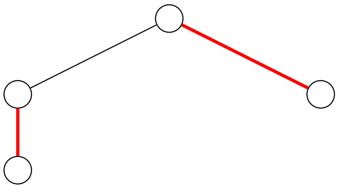
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



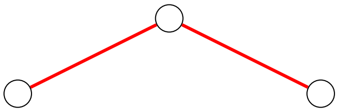
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



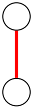
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees

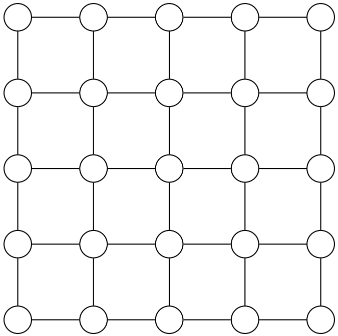


Cannot create a red degree-3 vertex

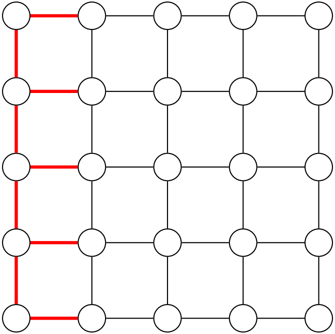
Graphs with bounded twin-width – trees

Generalization to bounded treewidth and even bounded rank-width

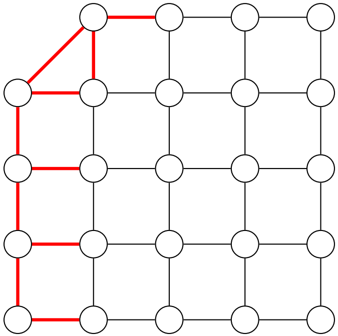
Graphs with bounded twin-width – grids



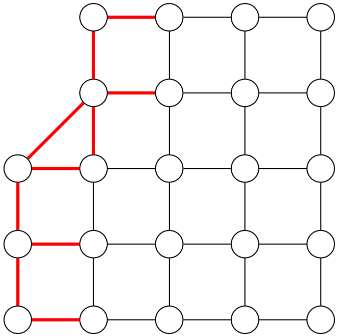
Graphs with bounded twin-width – grids



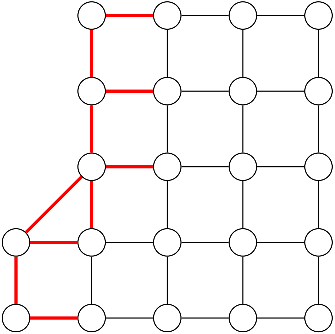
Graphs with bounded twin-width – grids



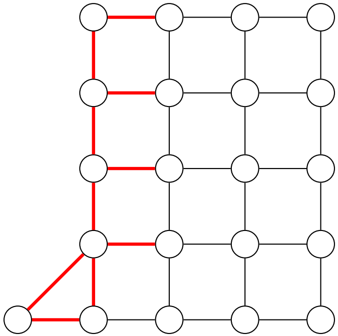
Graphs with bounded twin-width – grids



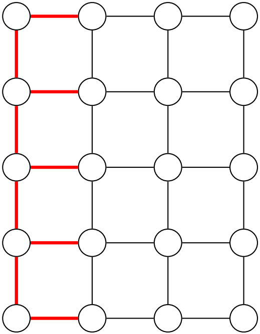
Graphs with bounded twin-width – grids



Graphs with bounded twin-width – grids



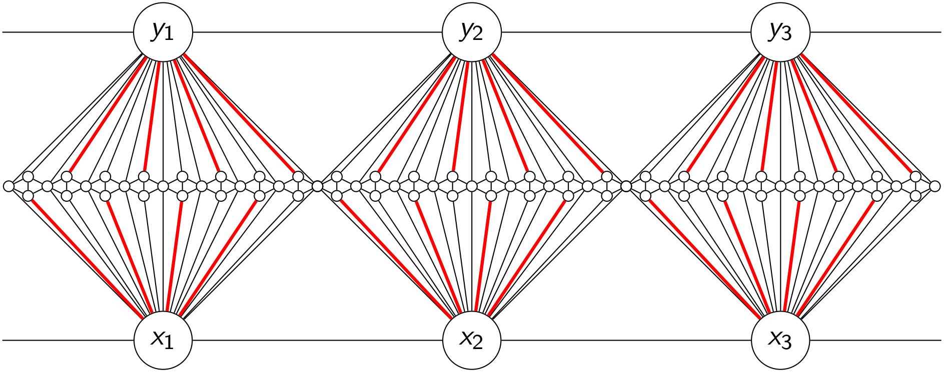
Graphs with bounded twin-width – grids



4-sequence for planar grids, $3d$-sequence for d-dimensional grids

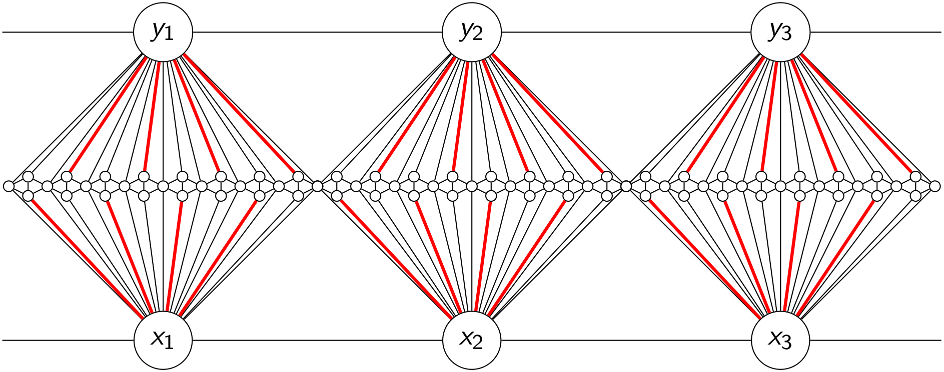
Graphs with bounded twin-width – planar graphs?

Graphs with bounded twin-width – planar graphs?



For every d, a planar trigraph without planar d-contraction

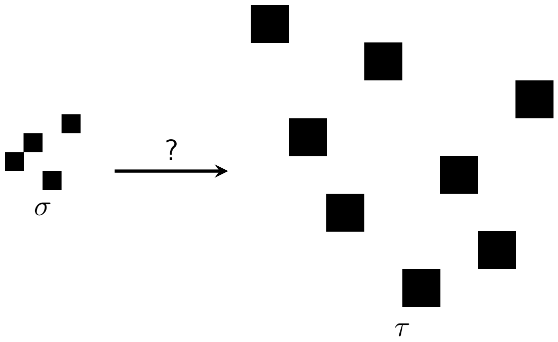
Graphs with bounded twin-width – planar graphs?



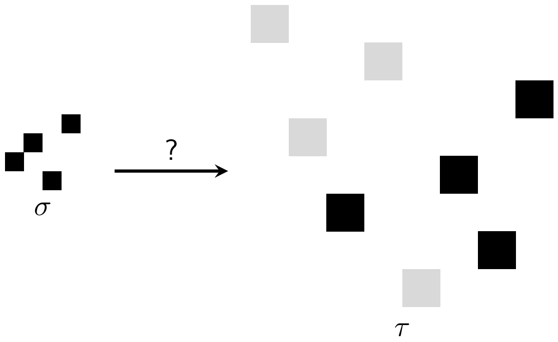
For every d, a planar trigraph without planar d-contraction

More powerfool tool needed

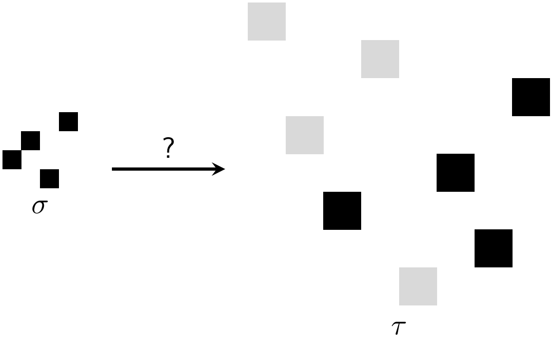
The origin: Permutation Pattern



The origin: PERMUTATION PATTERN



The origin: PERMUTATION PATTERN



Theorem (Guillemot, Marx '14)

PERMUTATION PATTERN *can be solved in time* $2^{|\sigma|^2}|\tau|$.

Guillemot and Marx's win-win algorithm

Theorem (Marcus, Tardos '04)

$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ entries 1 has a t-grid minor.

$$\text{4-grid minor} \quad \left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \hline
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

Guillemot and Marx's win-win algorithm

Theorem (Marcus, Tardos '04)

$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ entries 1 has a t-grid minor.

4-grid minor
$$\left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\hline
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
\hline
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

A) $\geqslant c_{|\sigma|} n$ entries 1 $\rightarrow$ YES from the $|\sigma|$-grid minor.
B) $< c_{|\sigma|} n$ entries 1 $\rightarrow$ merge of two "similar" rectangles

Guillemot and Marx's win-win algorithm

Theorem (Marcus, Tardos '04)

$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ entries 1 has a t-grid minor.

4-grid minor
$$\left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \hline
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

A) $\geqslant c_{|\sigma|} n$ entries 1 $\rightarrow$ YES from the $|\sigma|$-grid minor.
B) $< c_{|\sigma|} n$ entries 1 $\rightarrow$ merge of two "similar" rectangles

If B) always happens $\rightarrow$ DP on this merge sequence

Our generalization to the dense case – mixed minor

Mixed zone: not horizontal nor vertical

$$\left[\begin{array}{cc|ccc|ccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

3-mixed minor

Our generalization to the dense case – mixed minor

Mixed zone: not horizontal nor vertical

$$\left[\begin{array}{cc|ccc|ccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

3-mixed minor

A matrix is said t-**mixed free** if it does not have a t-mixed minor

Grid minor theorem for twin-width

Theorem (B, Kim, Thomassé, Watrigant 20)

If $\exists \sigma$ *s.t.* $Adj_\sigma(G)$ *is* t*-mixed free, then* $tww(G) = 2^{2^{O(t)}}$.

Grid minor theorem for twin-width

Theorem (B, Kim, Thomassé, Watrigant 20)

If $\exists\sigma$ *s.t.* $Adj_\sigma(G)$ *is* t*-mixed free, then* $tww(G) = 2^{2^{O(t)}}$*.*

Now to bound the twin-width of a class $\mathcal{C}$:

1) Find a *good* vertex-ordering procedure

2) Argue that, in this order, a t-mixed minor would conflict with $\mathcal{C}$

Grid minor theorem for twin-width

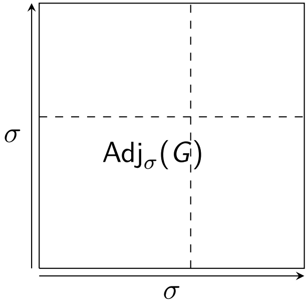
Theorem (B, Kim, Thomassé, Watrigant 20)

If $\exists\sigma$ *s.t.* $Adj_\sigma(G)$ *is* t*-mixed free, then* $tww(G) = 2^{2^{O(t)}}$.

Now to bound the twin-width of a class $\mathcal{C}$:

1) Find a *good* vertex-ordering procedure

2) Argue that, in this order, a t-mixed minor would conflict with $\mathcal{C}$



Cutting after the $t/2$-th division of the t-mixed minor

Grid minor theorem for twin-width

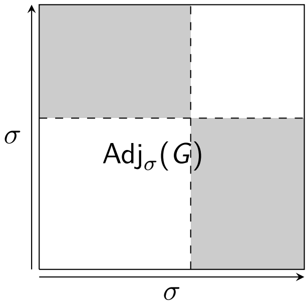
Theorem (B, Kim, Thomassé, Watrigant 20)

If $\exists\sigma$ *s.t.* $Adj_\sigma(G)$ *is* t*-mixed free, then* $tww(G) = 2^{2^{O(t)}}$.

Now to bound the twin-width of a class $\mathcal{C}$:

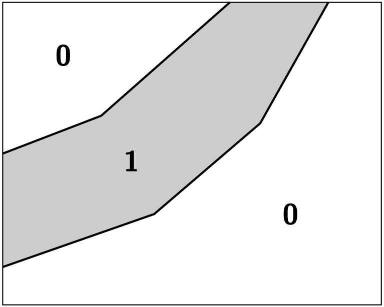
1) Find a *good* vertex-ordering procedure

2) Argue that, in this order, a t-mixed minor would conflict with $\mathcal{C}$



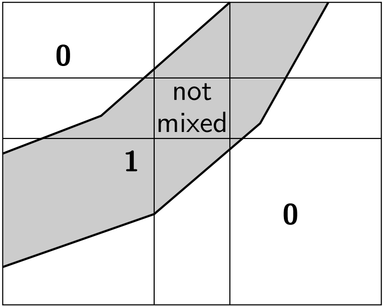
$t/2$-mixed minor on disjoint sets

Bounded twin-width – unit interval graphs



Warm-up with unit interval graphs: order by left endpoints

Bounded twin-width – unit interval graphs

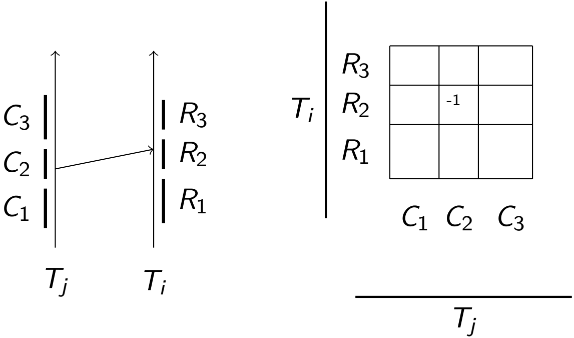


No 3-by-3 grid has all 9 cells crossed by two non-decreasing curves

Bounded twin-width – posets of bounded antichain

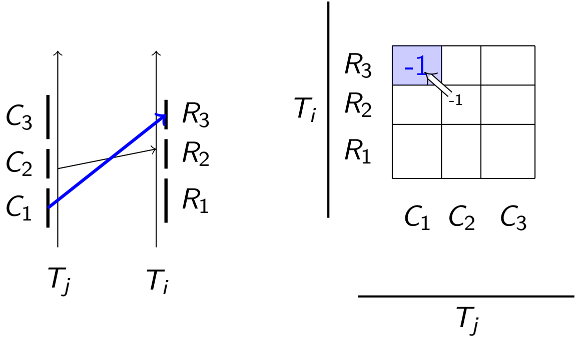
Put the k chains in order one after the other

Bounded twin-width – posets of bounded antichain



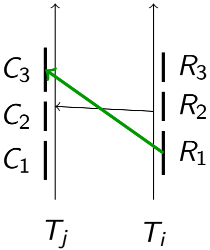
A $3k$-mixed minor implies a 3-mixed minor between two chains

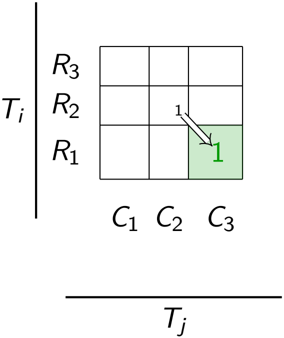
Bounded twin-width – posets of bounded antichain



Transitivity implies that a zone is constant

Bounded twin-width – posets of bounded antichain



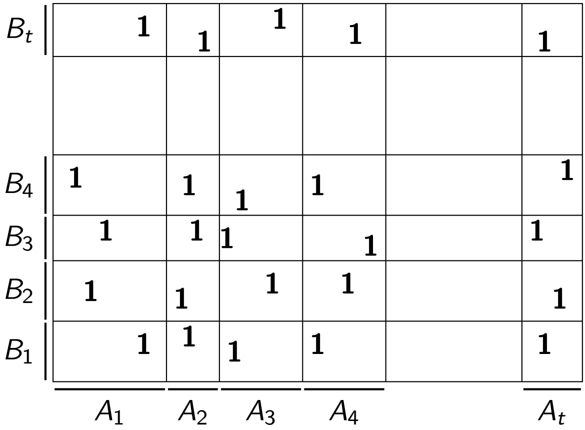


And symmetrically

Bounded twin-width – K_t-minor free graphs

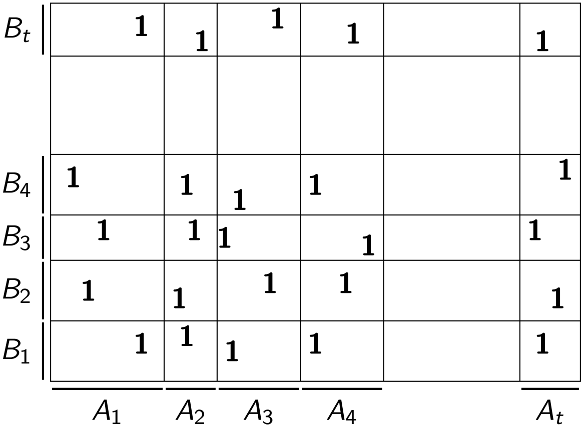
Given a hamiltonian path, we would just use this order

Bounded twin-width – K_t-minor free graphs



Contracting the $2t$ subpaths yields a $K_{t,t}$-minor, hence a K_t-minor

Bounded twin-width – K_t-minor free graphs



Instead we use a specially crafted lex-DFS discovery order

Theorem

The following classes have bounded twin-width, and $O(1)$-sequences can be computed in polynomial time.

- *Bounded rank-width, and even, boolean-width graphs,*
- *every hereditary proper subclass of permutation graphs,*
- *posets of bounded antichain size (seen as digraphs),*
- *unit interval graphs,*
- *K_t-minor free graphs,*
- *map graphs,*
- *subgraphs of d-dimensional grids,*
- *K_t-free unit d-dimensional ball graphs,*
- *$\Omega(\log n)$-subdivisions of all the n-vertex graphs,*
- *cubic expanders defined by iterative random 2-lifts from K_4,*
- *strong products of two bounded twin-width classes, one with bounded degree, etc.*

Theorem

The following classes have bounded twin-width, and $O(1)$-sequences can be computed in polynomial time.

- *Bounded rank-width, and even, boolean-width graphs,*
- *every hereditary proper subclass of permutation graphs,*
- *posets of bounded antichain size (seen as digraphs),*
- *unit interval graphs,*
- *K_t-minor free graphs,*
- *map graphs,*
- *subgraphs of d-dimensional grids,*
- *K_t-free unit d-dimensional ball graphs,*
- *$\Omega(\log n)$-subdivisions of all the n-vertex graphs,*
- *cubic expanders defined by iterative random 2-lifts from K_4,*
- *strong products of two bounded twin-width classes, one with bounded degree, etc.*

Can we solve problems faster, given an $O(1)$-sequence?

Example of k-Independent Set

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: **Compute by dynamic programming a best partial solution in each red connected subgraph of size at most k.**

Example of k-Independent Set

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: **Compute by dynamic programming a best partial solution in each red connected subgraph of size at most k.**

$d^{2k}n^2$ red connected subgraphs, actually only $d^{2k}n = 2^{O_d(k)}n$

Example of k-Independent Set

d-sequence: $G = G_n, G_{n-1}, \dots, G_2, G_1 = K_1$

Algorithm: **Compute by dynamic programming a best partial solution in each red connected subgraph of size at most k.**

$d^{2k} n^2$ red connected subgraphs, actually only $d^{2k} n = 2^{O_d(k)} n$

In G_n: red connected subgraphs are singletons, so are the solutions.
In G_1: If solution of size at least k, global solution.

Example of k-Independent Set

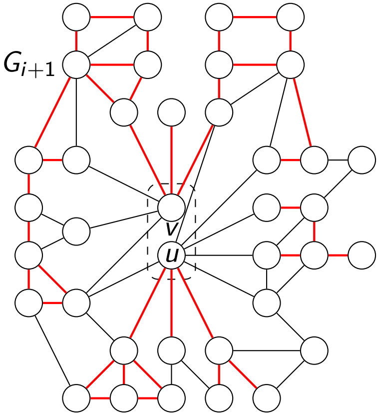
d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

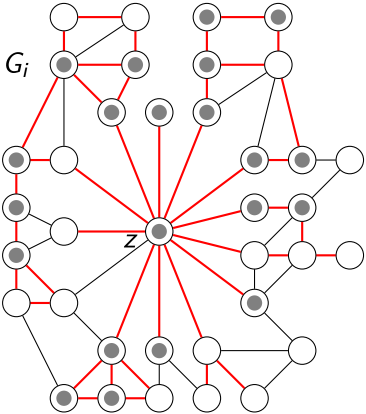
Algorithm: **Compute by dynamic programming a best partial solution in each red connected subgraph of size at most k.**

$d^{2k}n^2$ red connected subgraphs, actually only $d^{2k}n = 2^{O_d(k)}n$

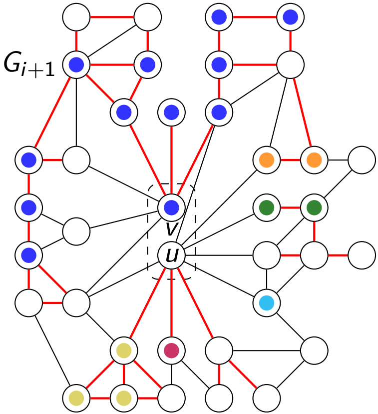
In G_n: red connected subgraphs are singletons, so are the solutions.
In G_1: If solution of size at least k, global solution.

How to go from the partial solutions of G_{i+1} to those of G_i?





Best partial solution inhabiting •?



3 unions of $\leqslant d+2$ red connected subgraphs to consider in G_{i+1} with u, or v, or both

Other (almost) single-exponential parameterized algorithms

Theorem

Given a d-sequence $G = G_n, \ldots, G_1 = K_1$,

- k-Independent Set,
- k-Clique,
- (r, k)-Scattered Set,
- k-Dominating Set, *and*
- (r, k)-Dominating Set

can be solved in time $2^{O_d(k)}n$,
whereas Subgraph Isomorphism *and* Induced Subgraph Isomorphism *can be solved in time $2^{O_d(k \log k)}n$.*

Other (almost) single-exponential parameterized algorithms

Theorem

Given a d-sequence $G = G_n, \ldots, G_1 = K_1$,

- k-Independent Set,
- k-Clique,
- (r, k)-Scattered Set,
- k-Dominating Set, *and*
- (r, k)-Dominating Set

can be solved in time $2^{O_d(k)} n$,
whereas Subgraph Isomorphism *and* Induced Subgraph Isomorphism *can be solved in time* $2^{O_d(k \log k)} n$.

A more general FPT algorithm?

First-order model checking on graphs

GRAPH FO MODEL CHECKING	**Parameter:** $\lvert\varphi\rvert$

Input: A graph G and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$
Question: $G \models \varphi$?

First-order model checking on graphs

GRAPH FO MODEL CHECKING	**Parameter:** $\lvert\varphi\rvert$
Input: A graph G and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$	
Question: $G \models \varphi$?	

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \forall x \bigvee_{1 \leqslant i \leqslant k} x = x_i \vee \bigvee_{1 \leqslant i \leqslant k} E(x, x_i) \vee E(x_i, x)$$

$G \models \varphi$? $\Leftrightarrow$

First-order model checking on graphs

GRAPH FO MODEL CHECKING	**Parameter:** $\lvert\varphi\rvert$
Input: A graph G and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$	
Question: $G \models \varphi$?	

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \forall x \bigvee_{1 \leqslant i \leqslant k} x = x_i \vee \bigvee_{1 \leqslant i \leqslant k} E(x, x_i) \vee E(x_i, x)$$

$G \models \varphi$? $\Leftrightarrow$ k-DOMINATING SET

First-order model checking on graphs

GRAPH FO MODEL CHECKING	**Parameter:** $\lvert\varphi\rvert$
Input: A graph G and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$	
Question: $G \models \varphi$?	

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \bigwedge_{1 \leqslant i < j \leqslant k} \neg(x_i = x_j) \wedge \neg E(x_i, x_j) \wedge \neg E(x_j, x_i)$$

$G \models \varphi$? $\Leftrightarrow$

First-order model checking on graphs

GRAPH FO MODEL CHECKING	**Parameter:** $\lvert\varphi\rvert$
Input: A graph G and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$	
Question: $G \models \varphi$?	

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \bigwedge_{1 \leqslant i < j \leqslant k} \neg(x_i = x_j) \wedge \neg E(x_i, x_j) \wedge \neg E(x_j, x_i)$$

$G \models \varphi$? $\Leftrightarrow$ k-INDEPENDENT SET

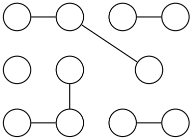
FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

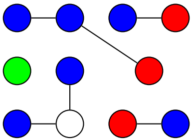
FO transduction: color by $O(1)$ unary relations, interpret, delete



FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete



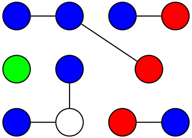
FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula

$\varphi(x, y) = \neg E(x, y)$ (complement)

$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete



$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg\exists z R(z) \wedge E(y, z))$$
$$\vee(R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg\exists z B(z) \wedge E(y, z))$$

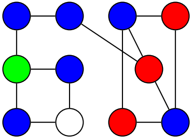
FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula

$\varphi(x, y) = \neg E(x, y)$ (complement)

$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete

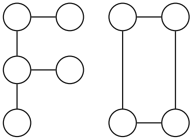


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg \exists z R(z) \wedge E(y, z))$$
$$\vee (R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg \exists z B(z) \wedge E(y, z))$$

FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

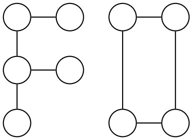
FO transduction: color by $O(1)$ unary relations, interpret, delete



FO interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete



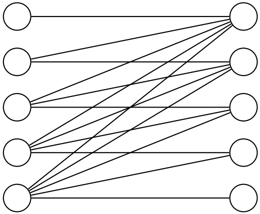
Theorem (B, Kim, Thomassé, Watrigant '20)

Bounded twin-width is preserved by transduction.

Monadically Stable and NIP

Stable class: no transduction of the class contains all ladders
NIP class: no transduction of the class contains all graphs

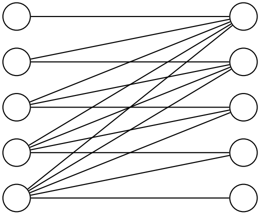
ladder



Monadically Stable and NIP

Stable class: no transduction of the class contains all ladders
NIP class: no transduction of the class contains all graphs

ladder

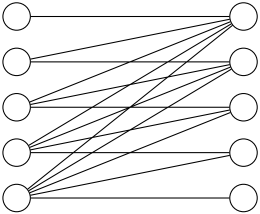


Bounded-degree graphs $\rightarrow$ stable
Unit interval graphs $\rightarrow$ NIP but not stable
Interval graphs $\rightarrow$ not NIP (triple negation!)

Monadically Stable and NIP

Stable class: no transduction of the class contains all ladders
NIP class: no transduction of the class contains all graphs

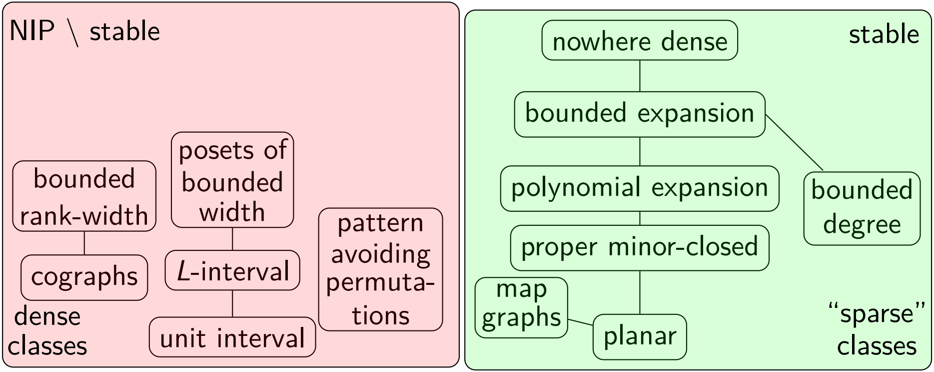
ladder



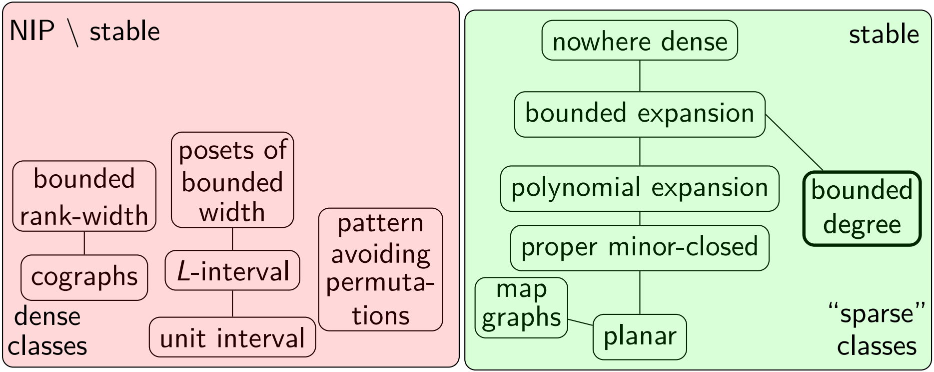
Bounded-degree graphs → stable
Unit interval graphs → NIP but not stable
Interval graphs → not NIP (triple negation!)

Bounded twin-width classes → NIP but not stable in general

Classes with known tractable FO model checking

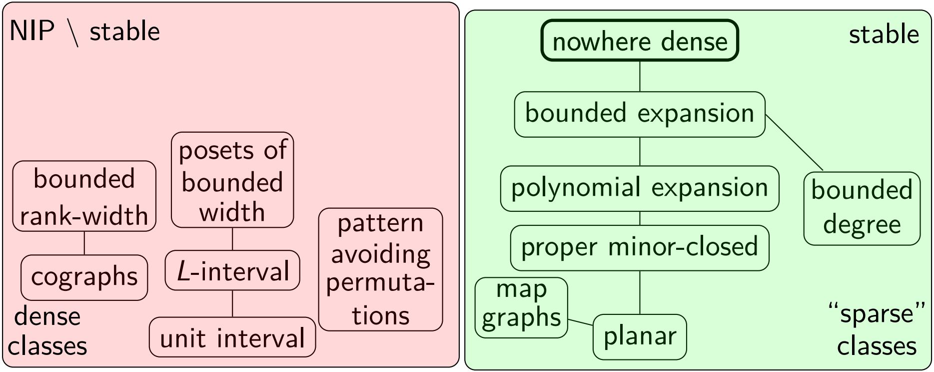


Classes with known tractable FO model checking



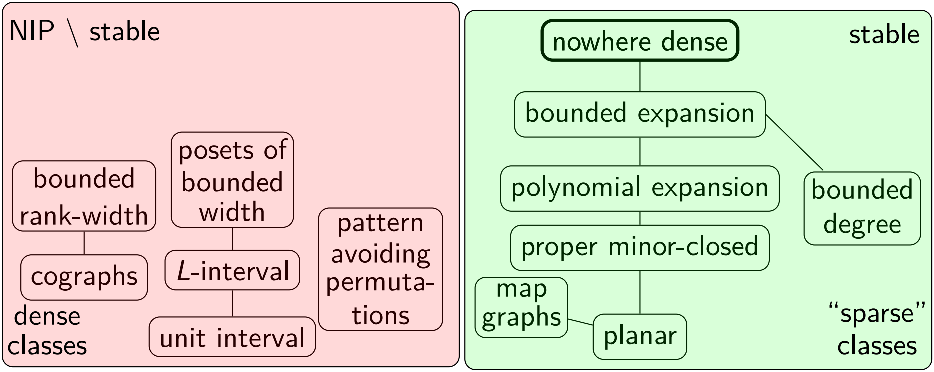
FO Model Checking solvable in $f(|\varphi|)n$ on bounded-degree graphs [Seese '96]

Classes with known tractable FO model checking



FO MODEL CHECKING solvable in $f(|\varphi|)n^{1+\varepsilon}$ on any nowhere dense class [Grohe, Kreutzer, Siebertz '14]

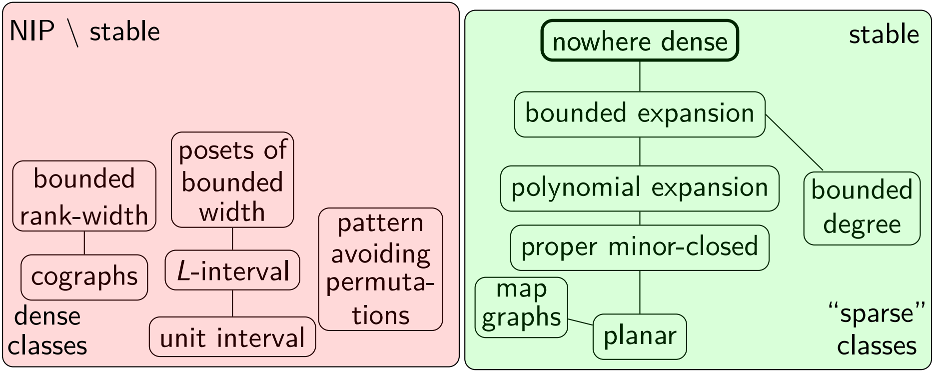
Classes with known tractable FO model checking



End of the story for the subgraph-closed classes

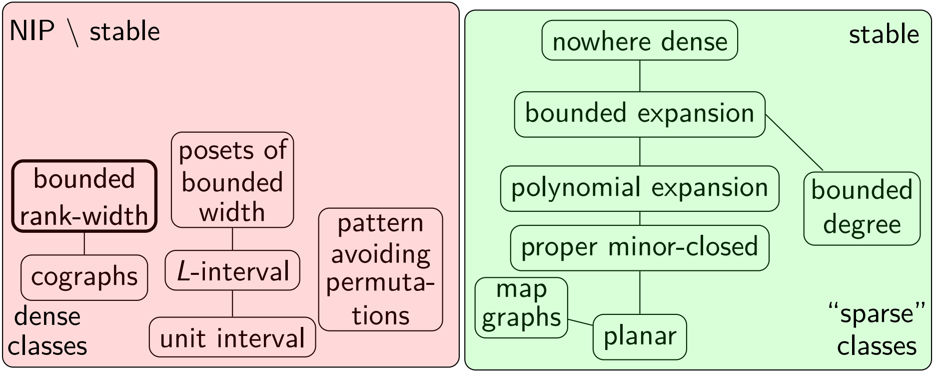
tractable FO Model Checking ⇔ nowhere dense ⇔ stable

Classes with known tractable FO model checking



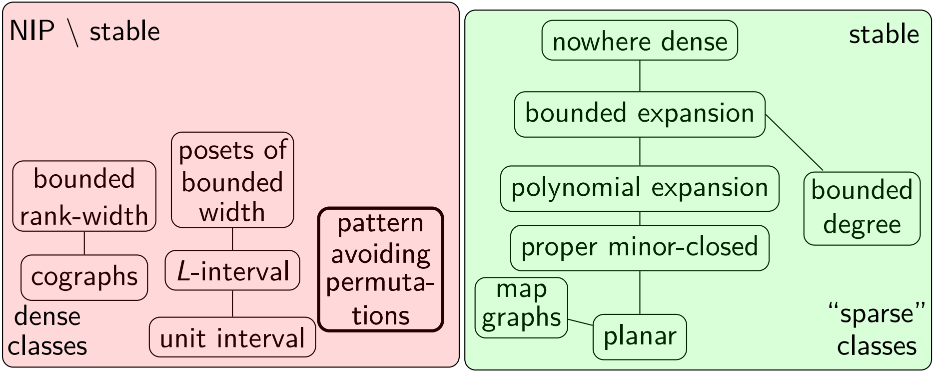
New program: transductions of nowhere dense classes
Not sparse anymore but still stable

Classes with known tractable FO model checking



MSO_1 Model Checking solvable in $f(|\varphi|, w)n$ on graphs of rank-width w
[Courcelle, Makowsky, Rotics '00]

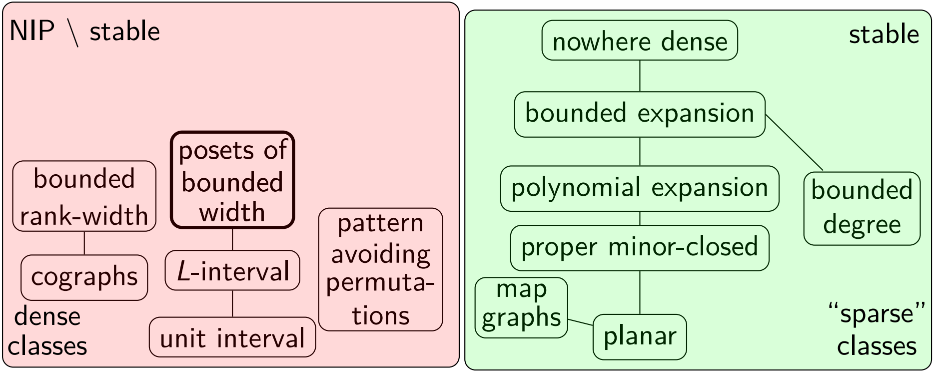
Classes with known tractable FO model checking



Is σ a subpermutation of τ? solvable in $f(|\sigma|)|\tau|$

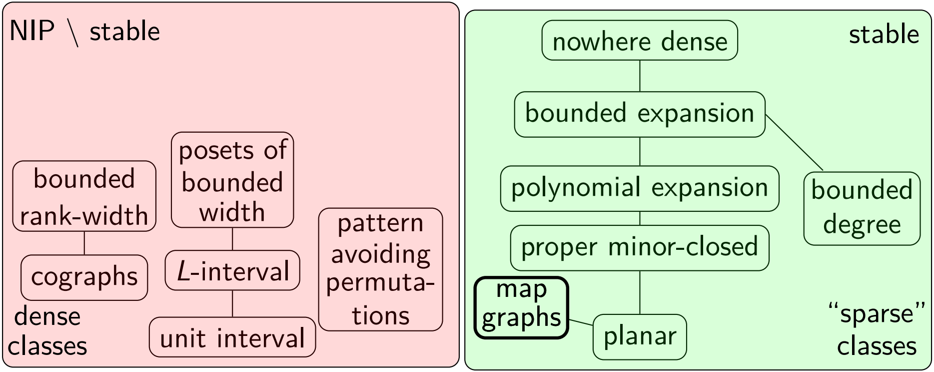
[Guillemot, Marx '14]

Classes with known tractable FO model checking



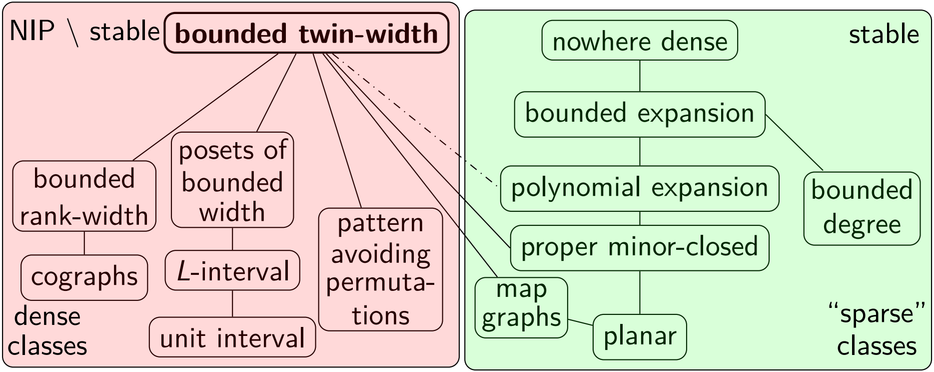
FO MODEL CHECKING solvable in $f(|\varphi|, w)n^2$ on posets of width w [GHLOORS '15]

Classes with known tractable FO model checking



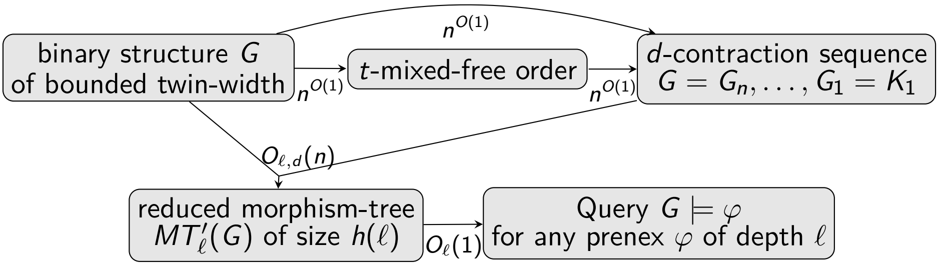
FO MODEL CHECKING solvable in $f(|\varphi|)n^{O(1)}$ on map graphs
[Eickmeyer, Kawarabayashi '17]

Classes with known tractable FO model checking

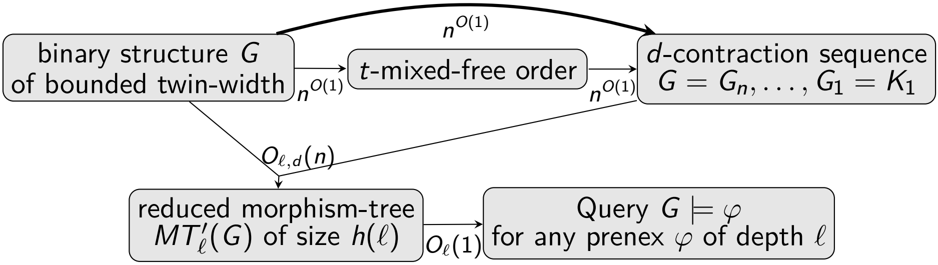


FO Model Checking solvable in $f(|\varphi|, d)n$ on graphs with a d-sequence [B, Kim, Thomassé, Watrigant '20]

Workflow of the FO model checking algorithm

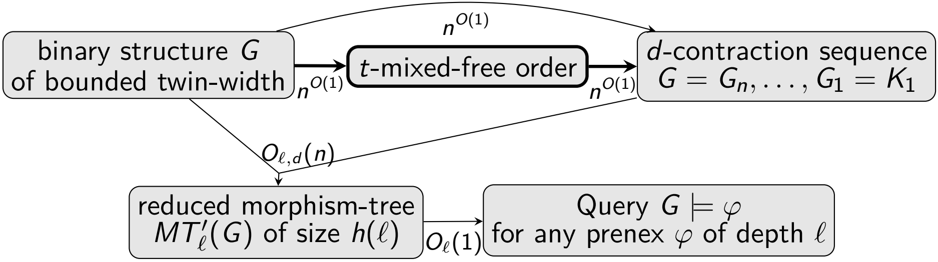


Workflow of the FO model checking algorithm



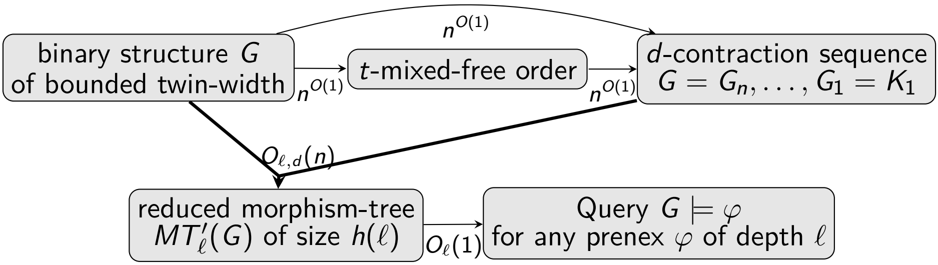
Direct examples: **trees**, bounded rank-width, **grids**, d-dimensional grids, unit interval graphs, K_t-free unit ball graphs

Workflow of the FO model checking algorithm



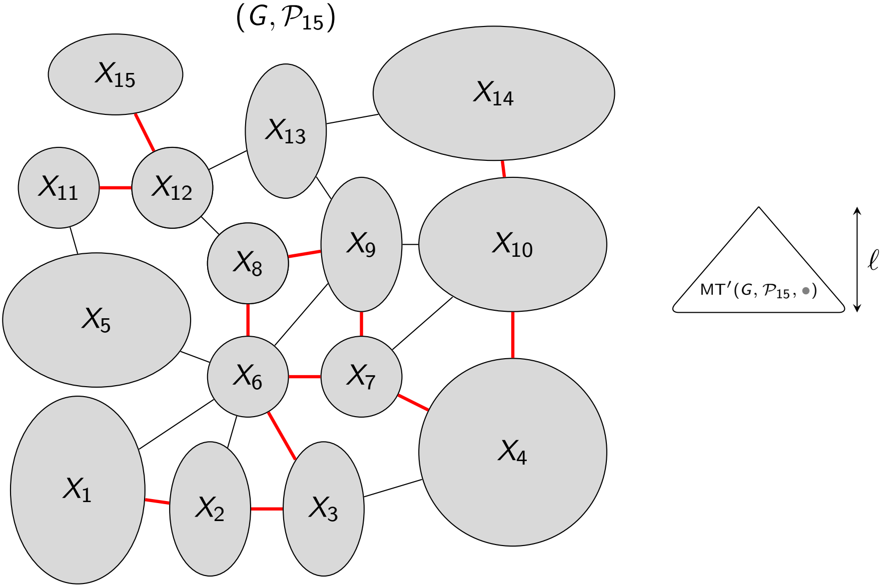
Detour via mixed minor for: pattern-avoiding permutations, **bounded width posets**, K_t-**minor free graphs**

Workflow of the FO model checking algorithm

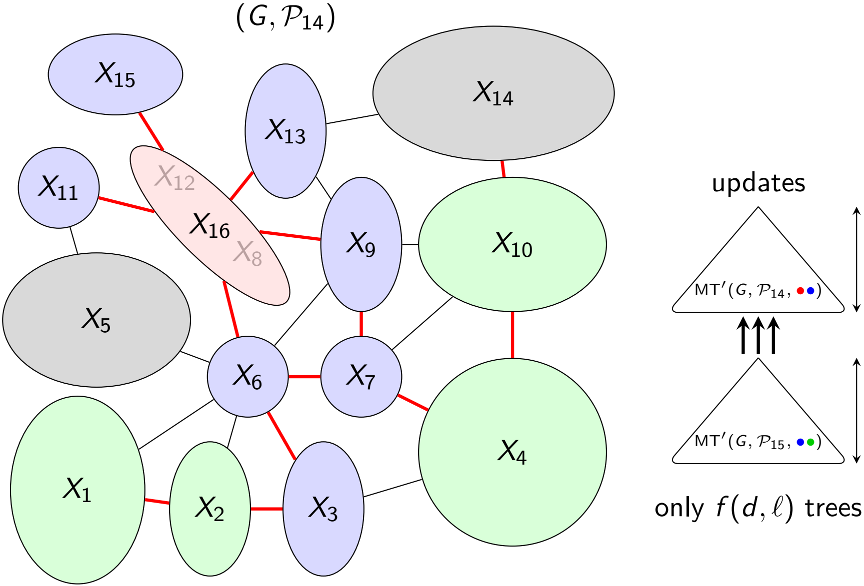


Let us see a snapshot of the FO model checking

DP for FO model checking with d-sequence



DP for FO model checking with d-sequence



Small classes

Small: class with at most $n!c^n$ labeled graphs on $[n]$.

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

Bounded twin-width classes are small.

Unifies and extends the same result for:
σ-free permutations [Marcus, Tardos '04]
K_t-minor free graphs [Norine, Seymour, Thomas, Wollan '06]

Small classes

Small: class with at most $n!c^n$ labeled graphs on $[n]$.

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

Bounded twin-width classes are small.

Subcubic graphs, interval graphs, triangle-free unit segment graphs have **unbounded** twin-width

Small classes

Small: class with at most $n!c^n$ labeled graphs on $[n]$.

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

Bounded twin-width classes are small.

Is the converse true for hereditary classes?

Conjecture (small conjecture)

A hereditary class has bounded twin-width if and only if it is small.

Sparse twin-width

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

If $\mathcal{C}$ is a hereditary class of bounded twin-width, tfae.

- *(i) $\mathcal{C}$ is $K_{t,t}$-free.*
- *(ii) $\mathcal{C}$ is d-grid free.*
- *(iii) Every n-vertex graph $G \in \mathcal{C}$ has at most gn edges.*
- *(iv) The subgraph closure of $\mathcal{C}$ has bounded twin-width.*
- *(v) $\mathcal{C}$ has bounded expansion.*

Sparse twin-width

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

If $\mathcal{C}$ is a hereditary class of bounded twin-width, tfae.

- *(i) $\mathcal{C}$ is $K_{t,t}$-free.*
- *(ii) $\mathcal{C}$ is d-grid free.*
- *(iii) Every n-vertex graph $G \in \mathcal{C}$ has at most gn edges.*
- *(iv) The subgraph closure of $\mathcal{C}$ has bounded twin-width.*
- *(v) $\mathcal{C}$ has bounded expansion.*

Still **fairly complicated:** bounded sparse twin-width classes comprise classes with bounded stack/queue number, flat classes, some particular expanders.

χ-boundedness

$\mathcal{C}$ χ-bounded: $\exists f, \forall G \in \mathcal{C}$, $\chi(G) \leqslant f(\omega(G))$

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

Every twin-width class is χ-bounded.
More precisely, every graph G of twin-width at most d admits a proper $(d+2)^{\omega(G)-1}$-coloring.

χ-boundedness

$\mathcal{C}$ χ-bounded: $\exists f, \forall G \in \mathcal{C}$, $\chi(G) \leqslant f(\omega(G))$

Theorem (B, Geniet, Kim, Thomassé, Watrigant 20+)

Every twin-width class is χ-bounded.
More precisely, every graph G of twin-width at most d admits a proper $(d+2)^{\omega(G)-1}$-coloring.

Polynomially χ-bounded? i.e., $\chi(G) = O(\omega(G)^d)$
At least strong Erdős-Hajnal property satisfied

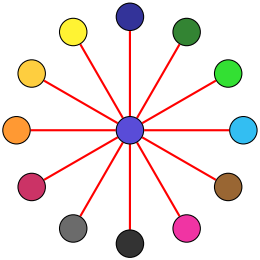
$d + 2$-coloring in the triangle-free case

Algorithm: **Start from $G_1 = K_1$, color its unique vertex 1, and rewind the d-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.**

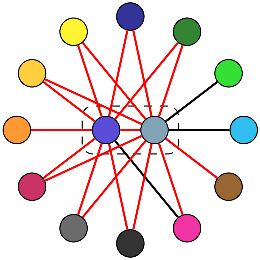
$d + 2$-coloring in the triangle-free case

Algorithm: **Start from $G_1 = K_1$, color its unique vertex 1, and rewind the d-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.**

$N_{G_i}[z]$



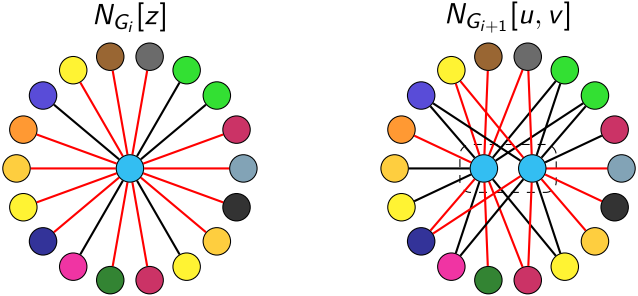
$N_{G_{i+1}}[u, v]$



z has only red incident edges $\rightarrow$ $d + 2$-nd color available to v

$d + 2$-coloring in the triangle-free case

Algorithm: **Start from $G_1 = K_1$, color its unique vertex 1, and rewind the d-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.**



z incident to at least one black edge $\rightarrow$ non-edge between u and v

Future directions

Obvious questions:
Algorithm to compute/approximate twin-width in general
Fully classify classes with tractable FO model checking
Small conjecture, polynomial expansion

Future directions

Obvious questions:
Algorithm to compute/approximate twin-width in general
Fully classify classes with tractable FO model checking
Small conjecture, polynomial expansion

Other directions we are exploring:
Better approximation algorithms on bounded twin-width classes
Twin-width of Cayley graphs of finitely generated groups
⋮

Future directions

Obvious questions:
Algorithm to compute/approximate twin-width in general
Fully classify classes with tractable FO model checking
Small conjecture, polynomial expansion

Other directions we are exploring:
Better approximation algorithms on bounded twin-width classes
Twin-width of Cayley graphs of finitely generated groups
⋮

On arxiv
Twin-width I: tractable FO model checking [BKTW '20]
Twin-width II: small classes [BGKTW '20]
Twin-width III: Max Independent Set and Coloring [BGKTW '20]