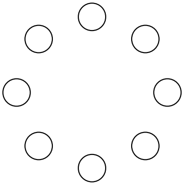
Twin-Width and Contraction Sequences

Édouard Bonnet

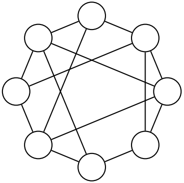
ENS de Lyon, LIP

April 19th 2024, Habilitation Defense, Lyon

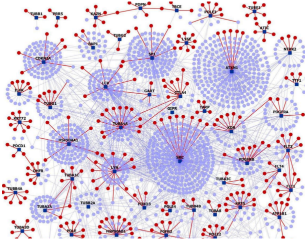
The modeling power of graphs

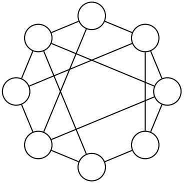


The modeling power of graphs

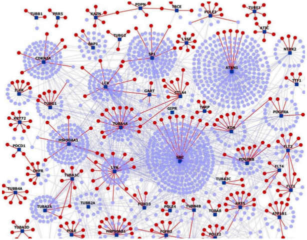


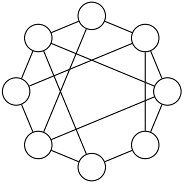
The modeling power of graphs



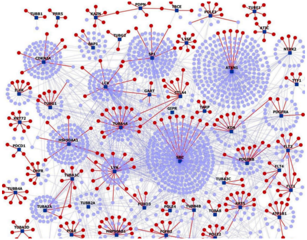


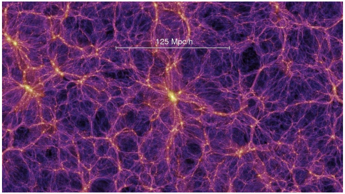
The modeling power of graphs

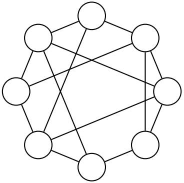




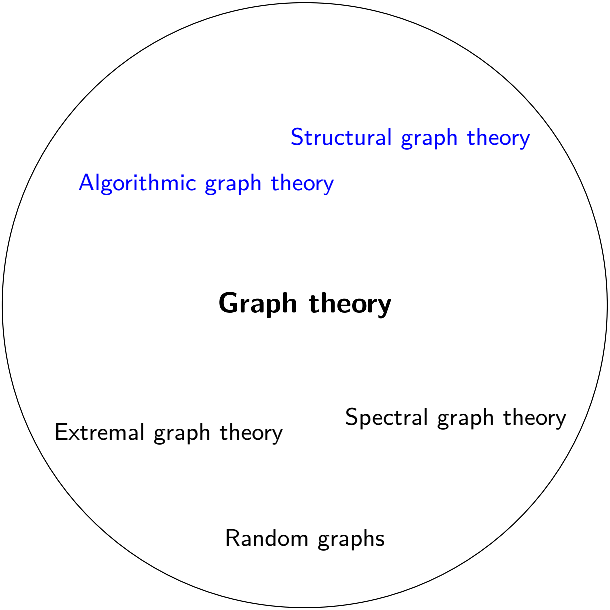
The modeling power of graphs



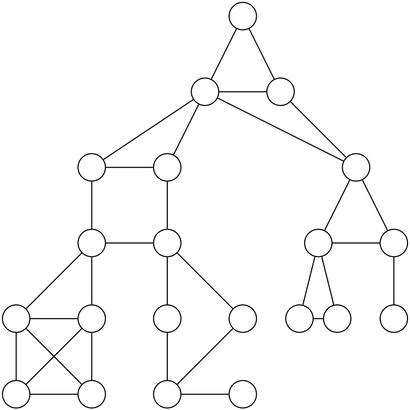




Graph theory and its interactions

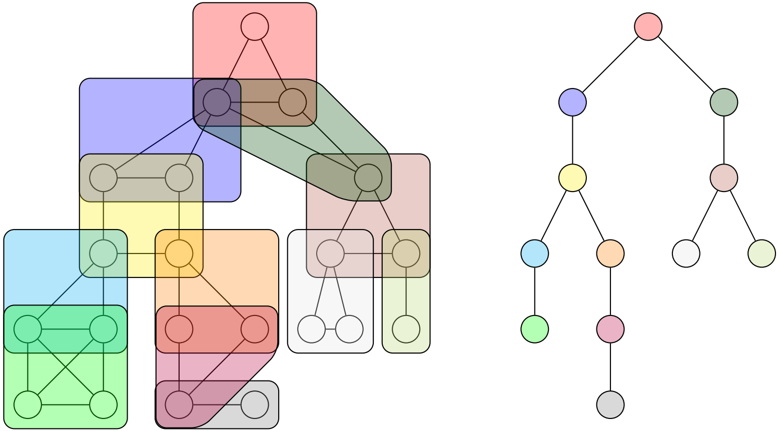


Tree-decomposition



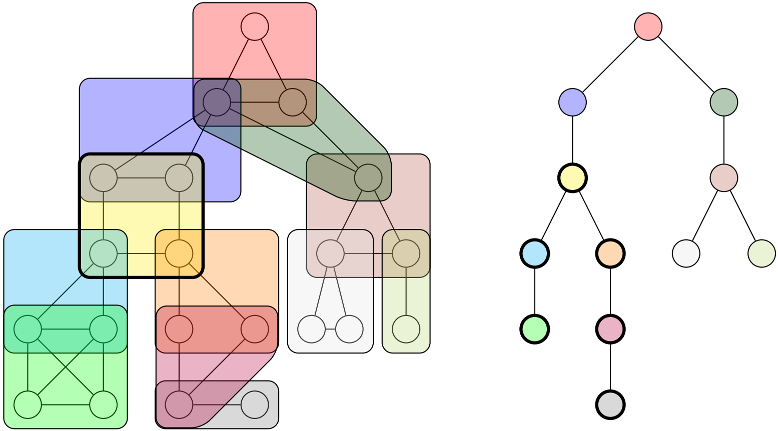
Tree-decomposition

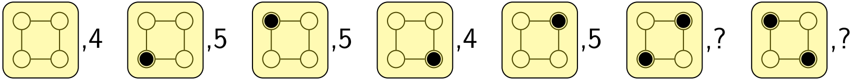
Cover by bags mapping to a tree s.t. each vertex lies in a subtree



Tree-decomposition: solving Max Independent Set

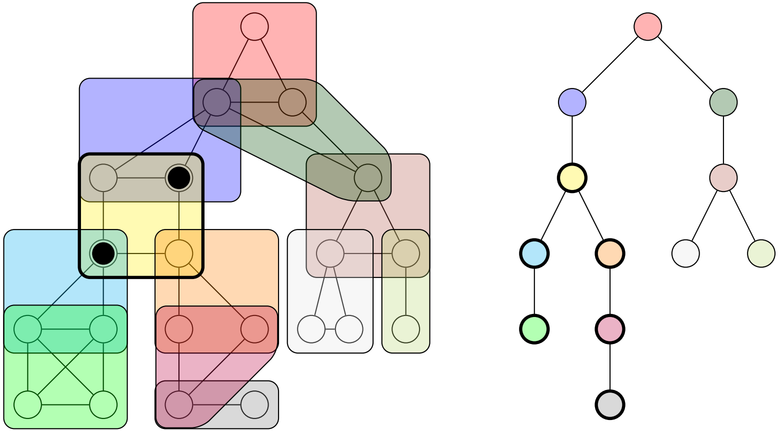
For each trace in each bag, keep a best solution in what is below

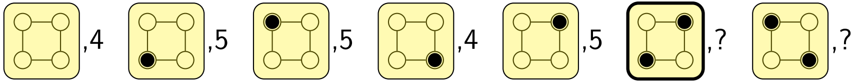




Tree-decomposition: solving Max Independent Set

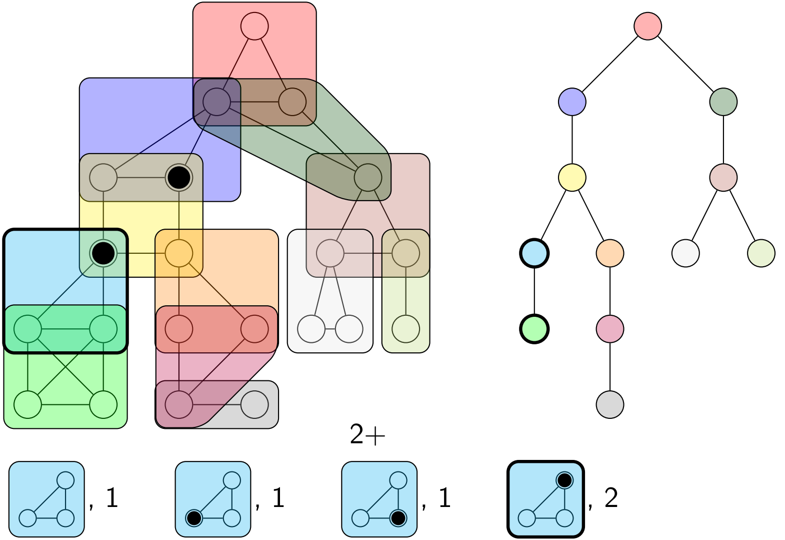
For each trace in each bag, keep a best solution in what is below





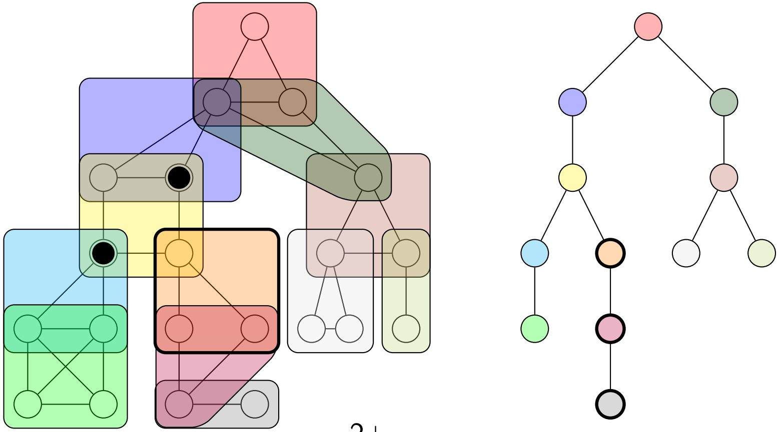
Tree-decomposition: solving Max Independent Set

For each trace in each bag, keep a best solution in what is below

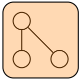


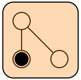
Tree-decomposition: solving Max Independent Set

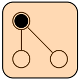
For each trace in each bag, keep a best solution in what is below

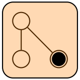


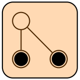
2+


, 1


, 2

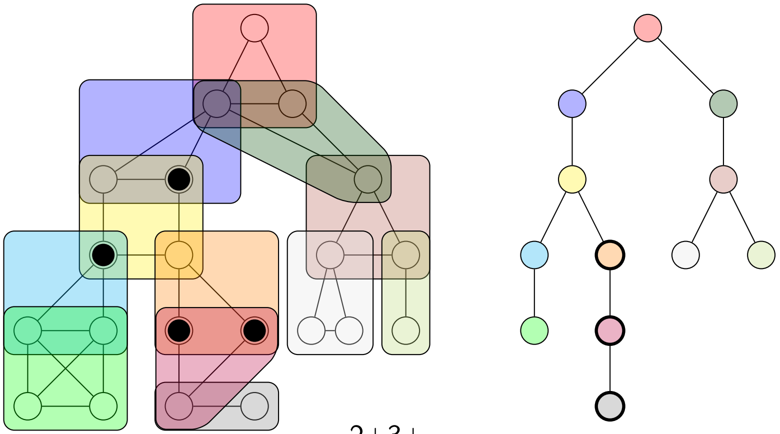

, 2


, 2

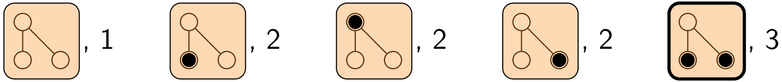

, 3

Tree-decomposition: solving Max Independent Set

For each trace in each bag, keep a best solution in what is below

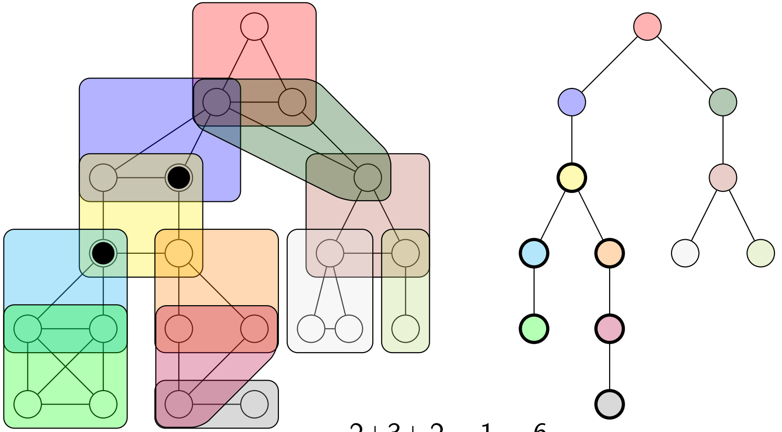


2+3+

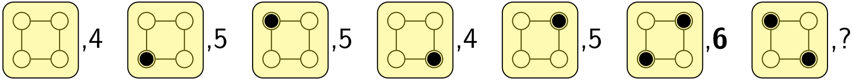


Tree-decomposition: solving Max Independent Set

For each trace in each bag, keep a best solution in what is below



$2+3+2-1=6$



Treewidth

Minimum largest bag size over all tree decompositions minus 1

- rediscovered several times in the 70's and 80's...
- made central by *Graph Minors* and algorithmic graph theory
- previous slide: $2^{O(\mathrm{tw})}n$ time for MAX INDEPENDENT SET
- generalized by Courcelle's theorem
- inspired other tree-based width parameters: clique-width, rank-width, mim-width, etc.

Treewidth

Minimum largest bag size over all tree decompositions minus 1

- rediscovered several times in the 70's and 80's...
- made central by *Graph Minors* and algorithmic graph theory
- previous slide: $2^{O(\mathrm{tw})} n$ time for MAX INDEPENDENT SET
- generalized by Courcelle's theorem
- inspired other tree-based width parameters: clique-width, rank-width, mim-width, etc.

Computing a tree decomposition?

Treewidth

Minimum largest bag size over all tree decompositions minus 1

- rediscovered several times in the 70's and 80's...
- made central by *Graph Minors* and algorithmic graph theory
- previous slide: $2^{O(\mathrm{tw})}n$ time for Max Independent Set
- generalized by Courcelle's theorem
- inspired other tree-based width parameters: clique-width, rank-width, mim-width, etc.

Computing a tree decomposition? NP-hard but various algorithms

width $2\mathrm{tw}+1$ in $2^{O(\mathrm{tw})}n$

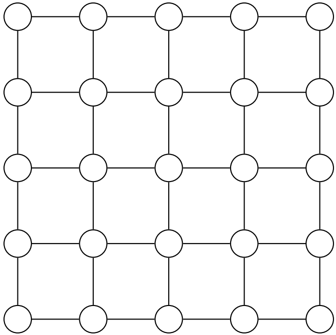
width $5\mathrm{tw}+4$ in $2^{6.76\mathrm{tw}}n\log n$

width tw in $2^{O(\mathrm{tw}^2)}n$

width $\mathrm{tw}\sqrt{\log \mathrm{tw}}$ in $n^{O(1)}$

width tw in 1.74^n

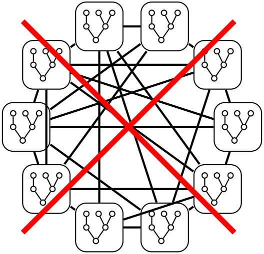
Low treewidth is very restrictive



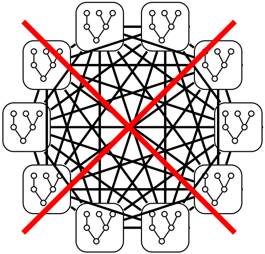
Grids have unbounded treewidth, clique-width, rank-width

Sparsity theory

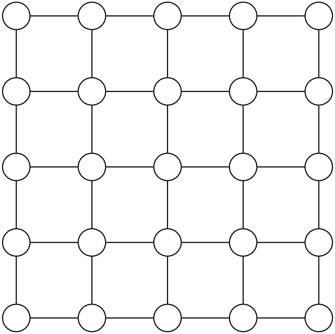
Bounded expansion: only sparse graph by shallow tree contraction

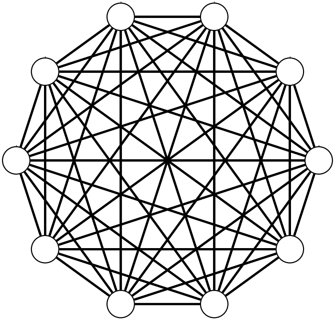


Nowhere denseness: no large clique by shallow tree contraction



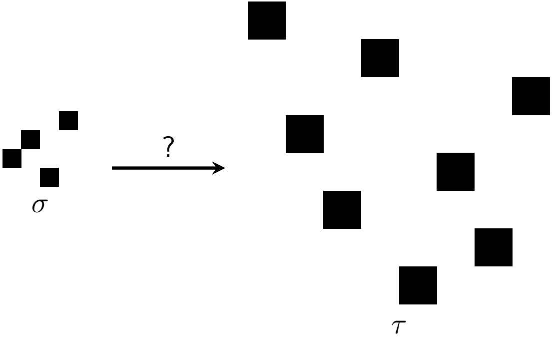
Going beyond sparsity and bounded clique-width?





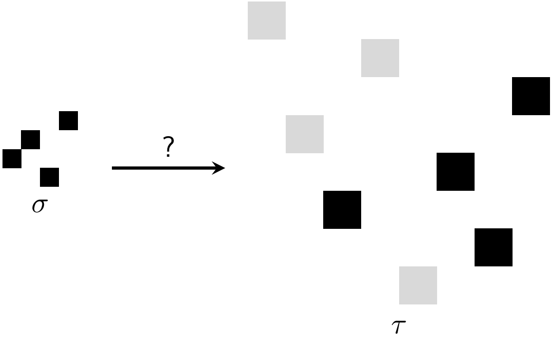
Conciliating the grid and the clique

The genesis of twin-width: Permutation Pattern



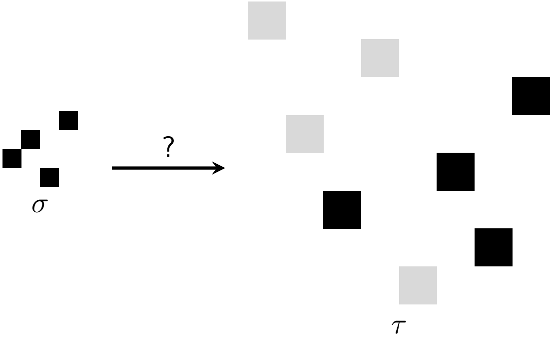
Is 3124 in 57362841?

The genesis of twin-width: Permutation Pattern



Is 3124 in 5**736**2**8**41? Yes

The genesis of twin-width: Permutation Pattern



Theorem (Guillemot, Marx '14)

Permutation Pattern *can be solved in time* $f(|\sigma|)|\tau|$.

Guillemot and Marx's win-win algorithm

Is σ in τ?

Theorem (Marcus, Tardos '04)

$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ 1-entries 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

Is σ in τ?

Theorem (Marcus, Tardos '04)

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

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

$\geqslant c_{|\sigma|} n$ 1-entries: answer YES from the $|\sigma|$-grid minor, or
$< c_{|\sigma|} n$ 1-entries: merge of two "similar" rectangles of 1s

Guillemot and Marx's win-win algorithm

Is σ in τ?

Theorem (Marcus, Tardos '04)

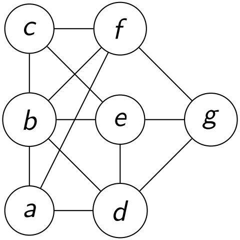
$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ 1-entries has a t-grid minor.

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

$\geqslant c_{|\sigma|} n$ 1-entries: answer YES from the $|\sigma|$-grid minor, or
$< c_{|\sigma|} n$ 1-entries: merge of two "similar" rectangles of 1s

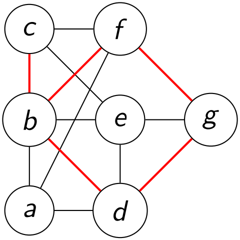
If the latter always holds: exploitable "decomposition" of τ

Graphs



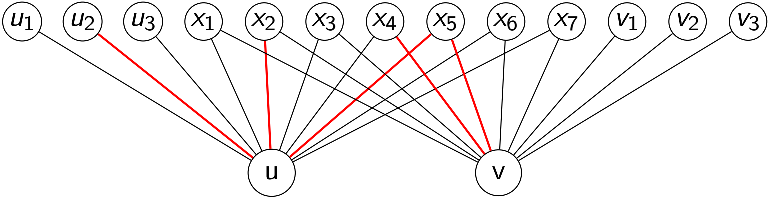
Two outcomes between a pair of vertices:
edge or non-edge

Trigraphs



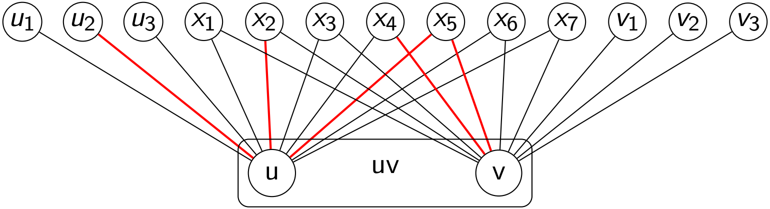
Three outcomes between a pair of vertices:
edge, or non-edge, or red edge (error edge)

Contractions in trigraphs



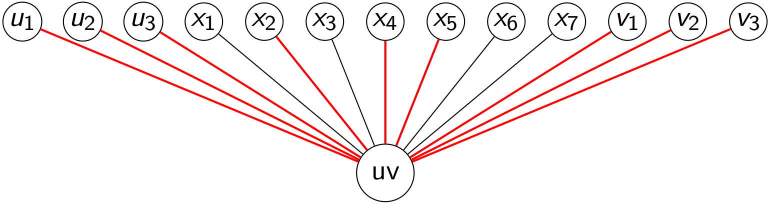
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



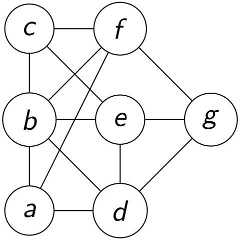
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



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

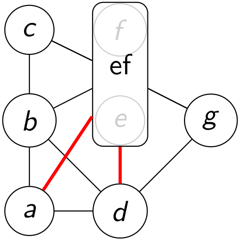
Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

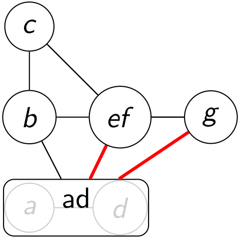
Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

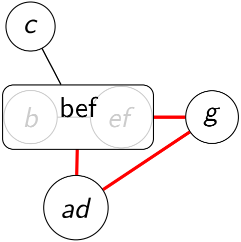
Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

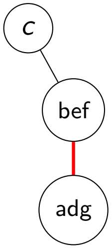
Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \dots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

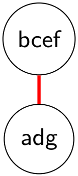
Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

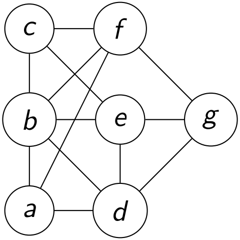
Contraction sequence

A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

$\mathcal{R}(G_i)$: red graph of G_i, obtained by removing its black edges

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.

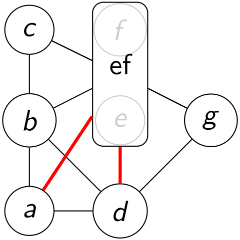


Maximum red degree $= 0$

overall maximum red degree = 0

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.

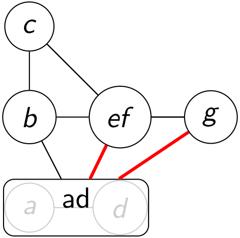


Maximum red degree $= 2$

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.

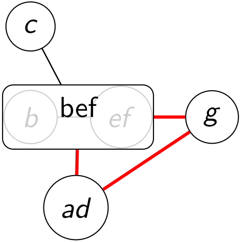


Maximum red degree $= 2$

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.

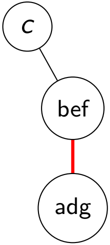


Maximum red degree = 2

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.

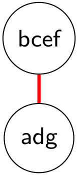


Maximum red degree $= 1$

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.



Maximum red degree $= 1$

overall maximum red degree = 2

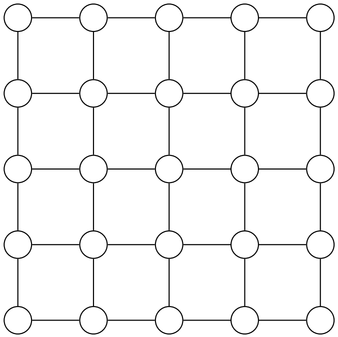
Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all red graphs have *maximum degree* at most d.

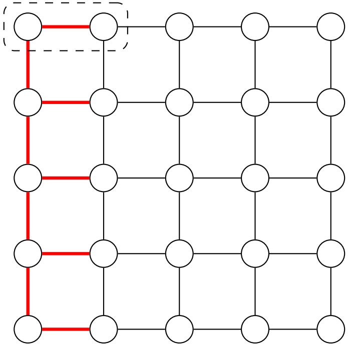
Maximum red degree $= 0$

overall maximum red degree = 2

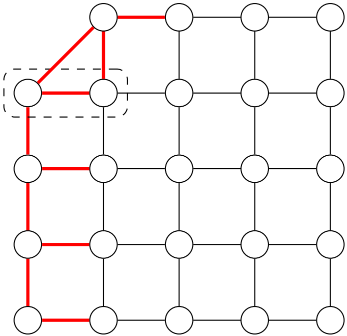
Grids have twin-width at most 4



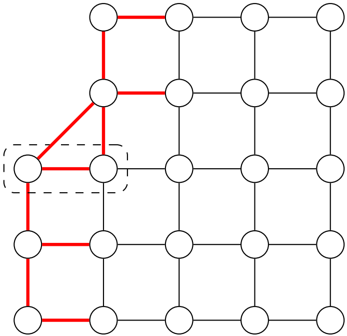
Grids have twin-width at most 4



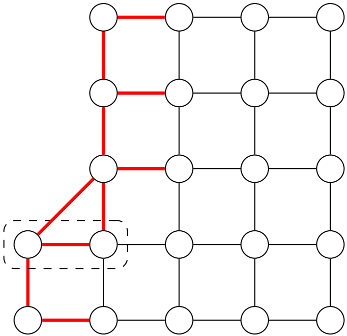
Grids have twin-width at most 4



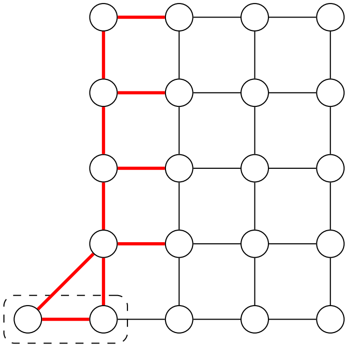
Grids have twin-width at most 4



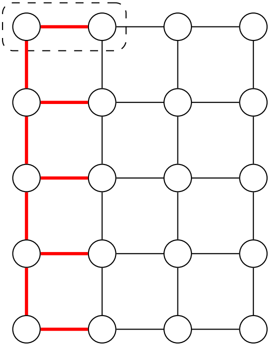
Grids have twin-width at most 4



Grids have twin-width at most 4

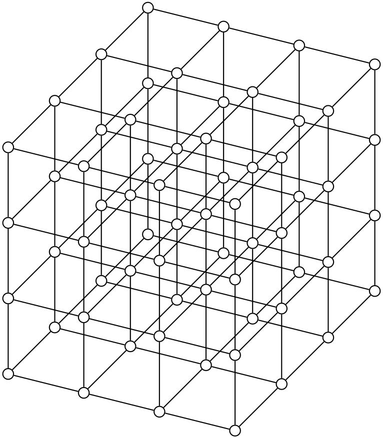


Grids have twin-width at most 4

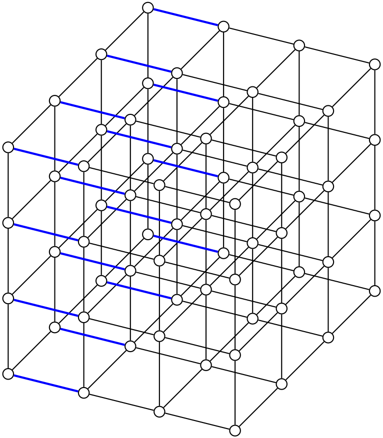


4-sequence for 2-dimensional grids

3-dimensional grids

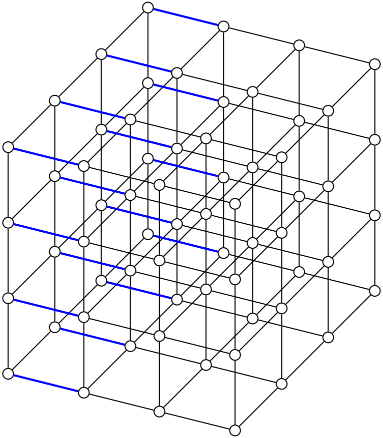


3-dimensional grids



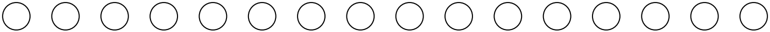
Contract the blue edges

3-dimensional grids



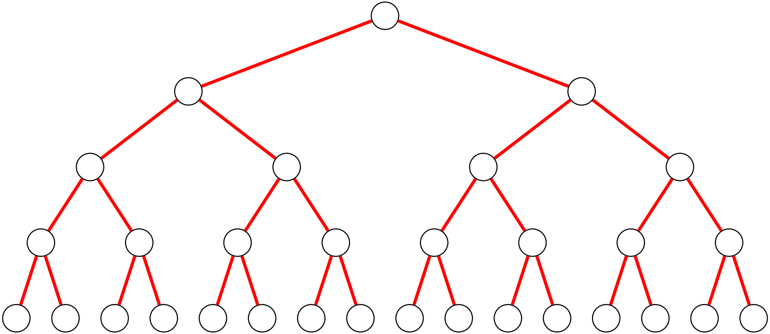
The d-dimensional grid has twin-width $\Theta(d)$

$(\geqslant 2\log n)$-subdivisions have twin-width at most 4



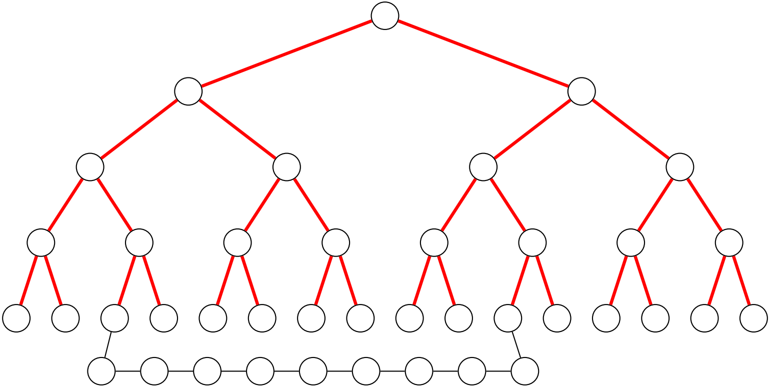
Consider the *branching* vertices

($\geqslant 2\log n$)-subdivisions have twin-width at most 4



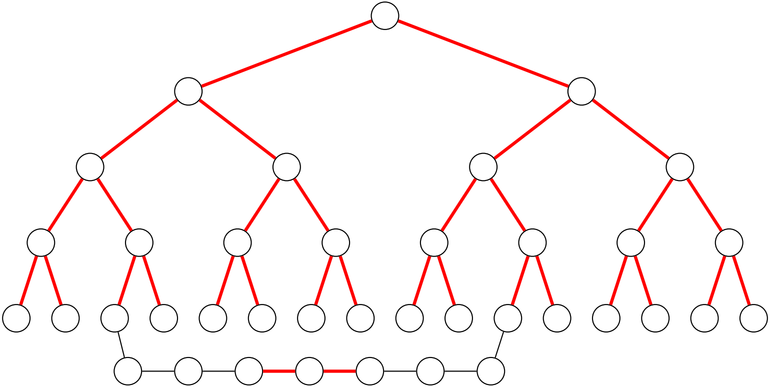
and make them leaves of a red full binary tree

($\geqslant 2 \log n$)-subdivisions have twin-width at most 4



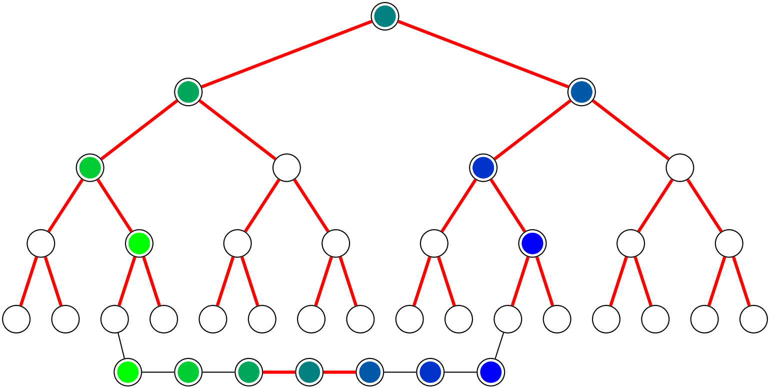
Take any subdivided edge

($\geqslant 2\log n$)-subdivisions have twin-width at most 4



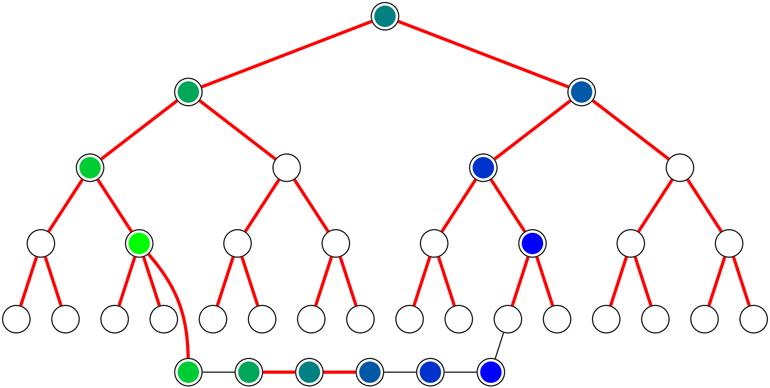
Shorten it to the length of the path in the red tree

($\geqslant 2 \log n$)-subdivisions have twin-width at most 4



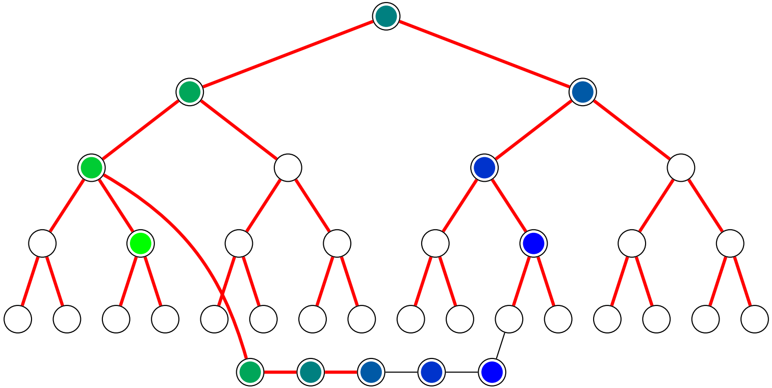
Zip the subdivided edge onto the tree

$(\geqslant 2 \log n)$-subdivisions have twin-width at most 4



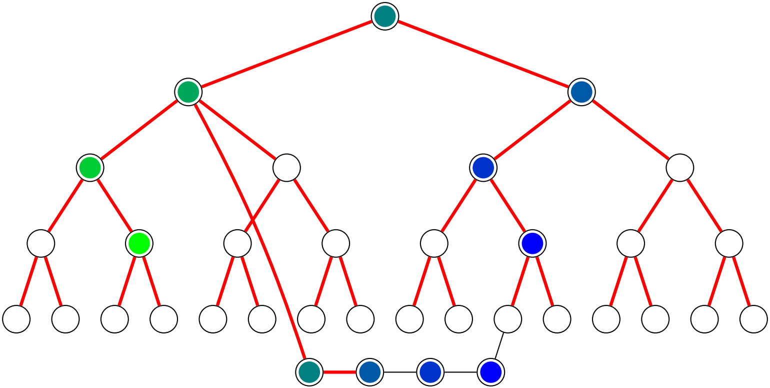
Zip the subdivided edge onto the tree

$(\geqslant 2\log n)$-subdivisions have twin-width at most 4



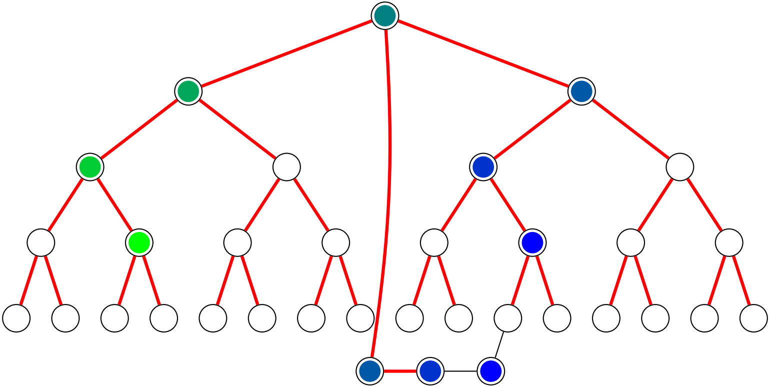
Zip the subdivided edge onto the tree

($\geqslant 2 \log n$)-subdivisions have twin-width at most 4



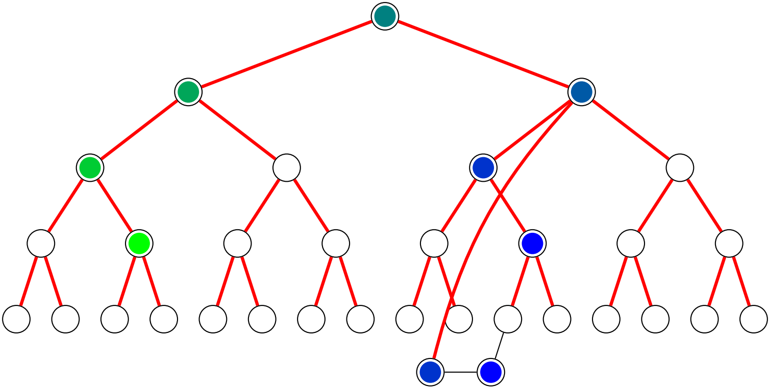
Zip the subdivided edge onto the tree

($\geqslant 2\log n$)-subdivisions have twin-width at most 4



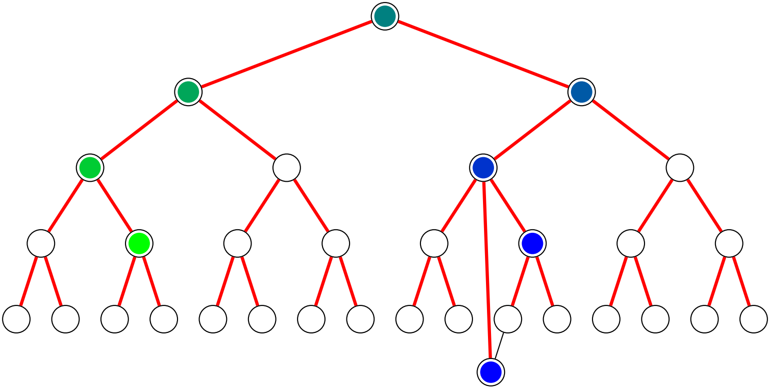
Zip the subdivided edge onto the tree

$(\geqslant 2\log n)$-subdivisions have twin-width at most 4



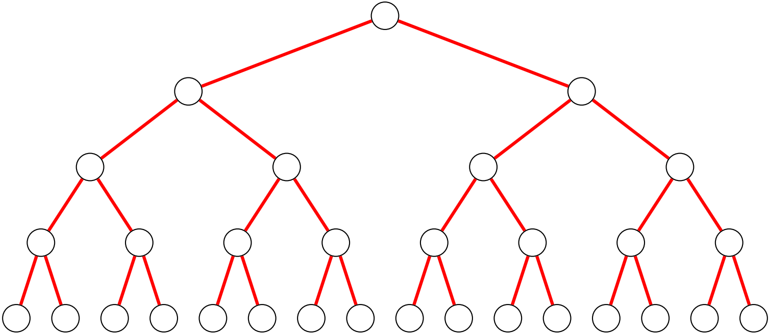
Zip the subdivided edge onto the tree

($\geqslant 2 \log n$)-subdivisions have twin-width at most 4



Zip the subdivided edge onto the tree

$(\geqslant 2 \log n)$-subdivisions have twin-width at most 4



Take another subdivided edge and repeat

Mixed minor

Mixed cell: at least two distinct rows and two distinct columns

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

3-mixed minor

Mixed minor

Mixed cell: at least two distinct rows and two distinct columns

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

3-mixed minor

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

Twin-width and mixed freeness

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

*If G admits **a** t-mixed free adjacency matrix, then* $tww(G) = 2^{2^{O(t)}}$.

Twin-width and mixed freeness

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

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

Twin-width and mixed freeness

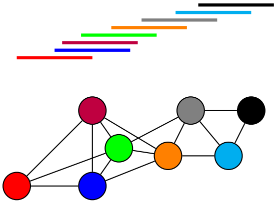
Theorem (B., Kim, Thomassé, Watrigant '20)

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

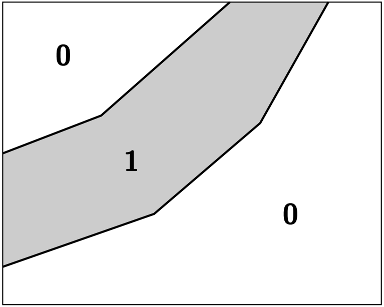
Now to bound the twin-width of a class $\mathcal{C}$:
1) Find a *good* vertex-ordering procedure
2) Argue that, in this order, a t-mixed minor would contradict the structure of $\mathcal{C}$

Unit interval graphs

Intersection graph of unit segments on the real line

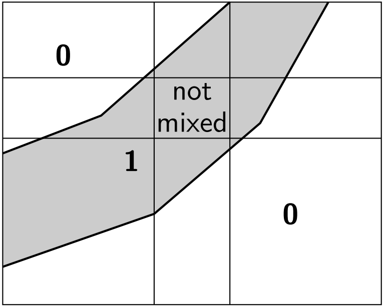


Unit interval graphs



order by left endpoints

Unit interval graphs



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

Classes known to have effective bounded twin-width

- 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.

Classes known to have effective bounded twin-width

- 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?

k-Independent Set

Algorithms in time $f(k)|V(G)|^{o(k)}$ are unlikely in general

Theorem (B., Geniet, Kim, Thomassé, Watrigant '21)

k-Independent Set can be solved in time $O(d^{2k}k^2|V(G)|)$ given a d-sequence $G = G_n, \ldots, G_1$.

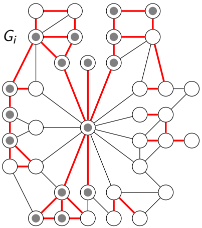
k-INDEPENDENT SET

Algorithms in time $f(k)|V(G)|^{o(k)}$ are unlikely in general

Theorem (B., Geniet, Kim, Thomassé, Watrigant '21)

k-INDEPENDENT SET can be solved in time $O(d^{2k}k^2|V(G)|)$ given a d-sequence $G = G_n, \ldots, G_1$.

Main idea: **For every $D \in \binom{V(G_i)}{\leqslant k}$ such that $\mathcal{R}(G_i)[D]$ is connected, store a largest independent set in $G[\bigcup D]$ intersecting every vertex of D, for i going to n down to 1.**



k-Independent Set: First observations

Main idea: **For every $D \in \binom{V(G_i)}{\leqslant k}$ such that $\mathcal{R}(G_i)[D]$ is connected, store a largest independent set in $G[\bigcup D]$ intersecting every vertex of D, for i going to n down to 1.**

k-Independent Set: First observations

Main idea: **For every $D \in \binom{V(G_i)}{\leqslant k}$ such that $\mathcal{R}(G_i)[D]$ is connected, store a largest independent set in $G[\bigcup D]$ intersecting every vertex of D, for i going to n down to 1.**

Initialization: Every connected set in $\mathcal{R}(G_n)$ is of the form $\{v\}$ for $v \in V(G)$, for which we store the independent set $\{v\}$.

k-Independent Set: First observations

Main idea: **For every $D \in \binom{V(G_i)}{\leqslant k}$ such that $\mathcal{R}(G_i)[D]$ is connected, store a largest independent set in $G[\bigcup D]$ intersecting every vertex of D, for i going to n down to 1.**

Initialization: Every connected set in $\mathcal{R}(G_n)$ is of the form $\{v\}$ for $v \in V(G)$, for which we store the independent set $\{v\}$.

Completeness: If no independent set of size at least k is detected, a maximum independent set of G is stored for $\{V(G)\}$ in $\mathcal{R}(G_1)$.

k-Independent Set: First observations

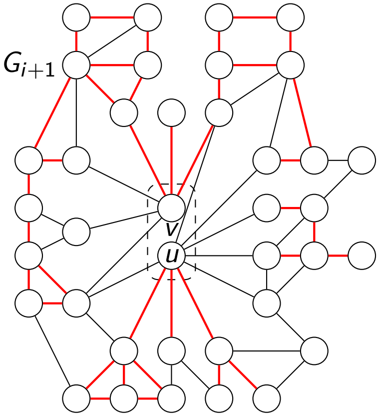
Main idea: **For every $D \in \binom{V(G_i)}{\leqslant k}$ such that $\mathcal{R}(G_i)[D]$ is connected, store a largest independent set in $G[\bigcup D]$ intersecting every vertex of D, for i going to n down to 1.**

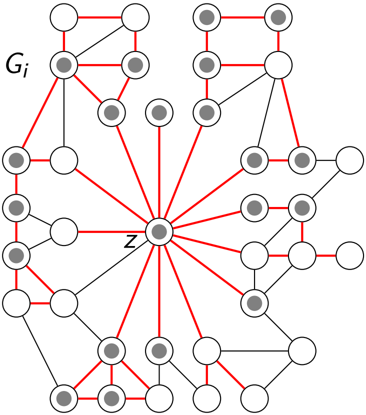
Initialization: Every connected set in $\mathcal{R}(G_n)$ is of the form $\{v\}$ for $v \in V(G)$, for which we store the independent set $\{v\}$.

Completeness: If no independent set of size at least k is detected, a maximum independent set of G is stored for $\{V(G)\}$ in $\mathcal{R}(G_1)$.

Running time: As $\mathcal{R}(G_i)$ has maximum degree at most d, it has at most $d^{2k}i$ connected sets on up to k vertices.

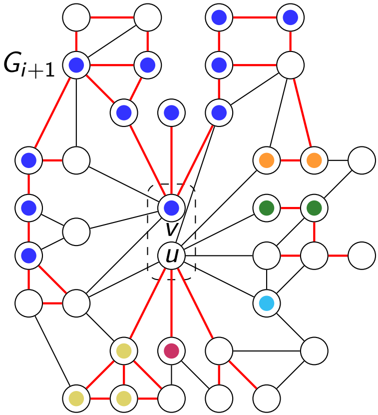
k-Independent Set: Update of partial solutions

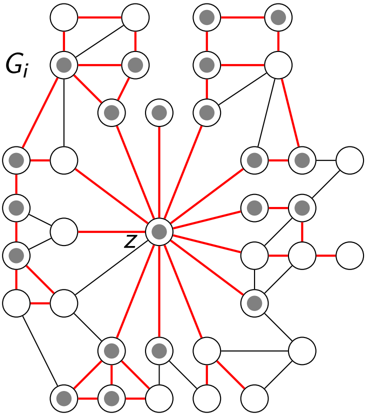




Best partial solution inhabiting •?

k-Independent Set: Update of partial solutions





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

Model checking

Graph FO/MSO Model Checking **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Model checking

Graph FO/MSO Model Checking **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

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

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

Model checking

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

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
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

Model checking

Graph FO/MSO Model Checking **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

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

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

Model checking

GRAPH FO/MSO MODEL CHECKING	**Parameter:** $\|\varphi\|$

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
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

Model checking

Graph FO/MSO Model Checking **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists X_1 \exists X_2 \exists X_3 (\forall x \bigvee_{1 \leqslant i \leqslant 3} X_i(x)) \land \forall x \forall y \bigwedge_{1 \leqslant i \leqslant 3} (X_i(x) \land X_i(y) \rightarrow \neg E(x, y))$$

$G \models \varphi? \Leftrightarrow$

Model checking

Graph FO/MSO Model Checking **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists X_1 \exists X_2 \exists X_3 (\forall x \bigvee_{1 \leqslant i \leqslant 3} X_i(x)) \wedge \forall x \forall y \bigwedge_{1 \leqslant i \leqslant 3} (X_i(x) \wedge X_i(y) \rightarrow \neg E(x,y))$$

$G \models \varphi$? $\Leftrightarrow$ 3-Coloring

Model checking

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

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

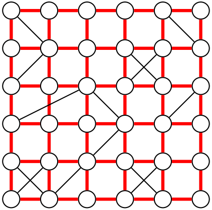
$$\varphi = \exists X_1 \exists X_2 \exists X_3 (\forall x \bigvee_{1 \leqslant i \leqslant 3} X_i(x)) \wedge \forall x \forall y \bigwedge_{1 \leqslant i \leqslant 3} (X_i(x) \wedge X_i(y) \rightarrow \neg E(x, y))$$

$G \models \varphi$? $\Leftrightarrow$ 3-COLORING

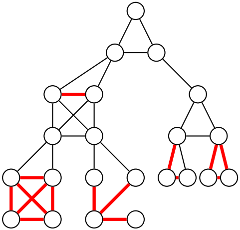
When can we solve MODEL CHECKING in time $f(\varphi)\lvert V(G)\rvert^{O(1)}$?

Reduced parameters and component twin-width

$$p^{\downarrow}(G) = \min\{\max_{i \in [n]} p(\mathcal{R}(G_i)) : G_n, \ldots, G_1 \text{ sequence of } G\}$$

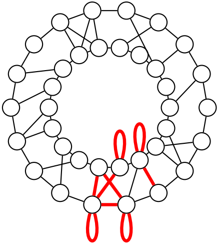


reduced maximum degree = twin-width



reduced component size $\equiv$ cliquewidth

= component twin-width



reduced #edges $\equiv$ linear cliquewidth

Model checking on graphs of bounded twin-width

Recast of the Courcelle–Makowsky–Rotics theorem:

Theorem (B., Kim, Reinald, Thomassé '22)

MSO model checking can be solved in time $f(|\varphi|, d) \cdot |V(G)|$ on graphs G given with a d-sequence of component twin-width.

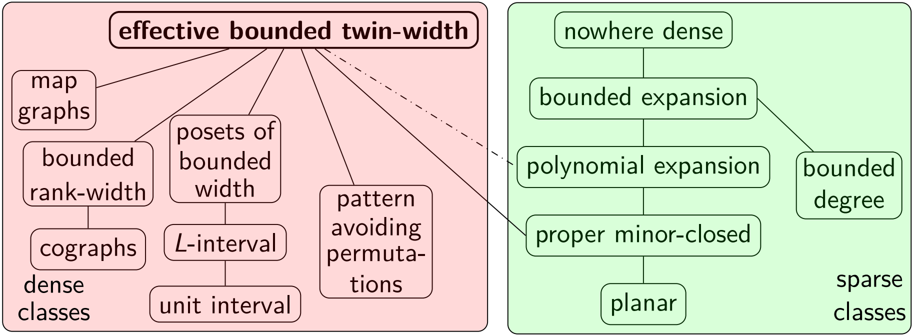
Generalization of the k-Independent Set algorithm:

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

FO model checking can be solved in time $f(|\varphi|, d) \cdot |V(G)|$ on graphs G given with a d-sequence.

Gaifman's locality + MSO model checking algorithm

Classes for which FO model checking is FPT



Theorem (Bergé, B., Déprés '22)

Deciding if the twin-width of a graph is at most 4 is NP-complete.

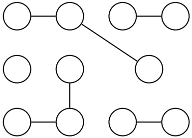
First-order interpretations and transductions

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)

First-order 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)

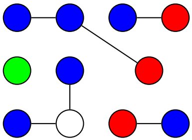
FO transduction: color by $O(1)$ unary relations, interpret, delete



First-order 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)

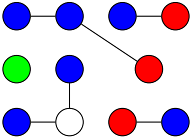
FO transduction: color by $O(1)$ unary relations, interpret, delete



First-order 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

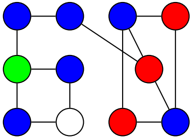


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg\exists z R(z) \wedge E(y, z))$$
$$\vee (R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg\exists z B(z) \wedge E(y, z))$$

First-order 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

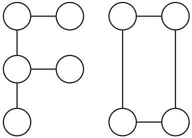


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg\exists z R(z) \wedge E(y, z))$$
$$\vee (R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg\exists z B(z) \wedge E(y, z))$$

First-order 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

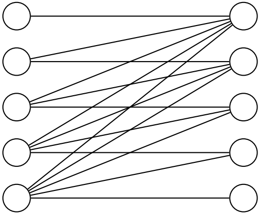


Stable and dependent (for hereditary classes)

Due to [Baldwin, Shelah '85; Braunfeld, Laskowski '22]

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

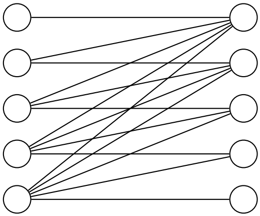
ladder



Stable and dependent (for hereditary classes)

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

ladder

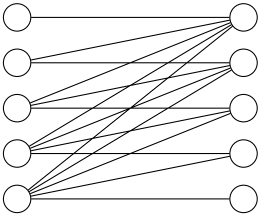


Bounded-degree graphs $\to$ stable
Unit interval graphs $\to$ dependent but not stable
Interval graphs $\to$ independent

Stable and dependent (for hereditary classes)

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

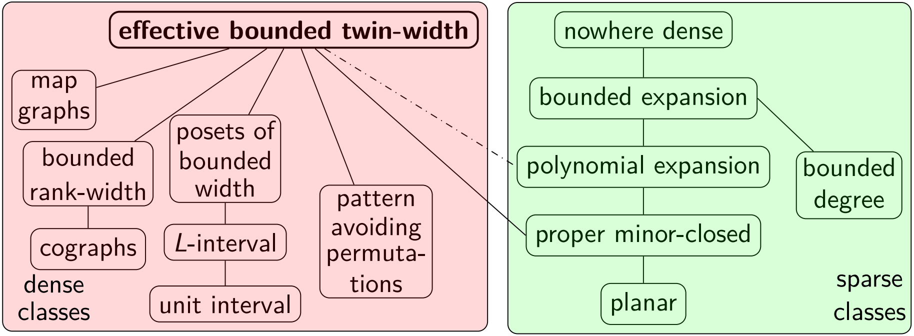
ladder



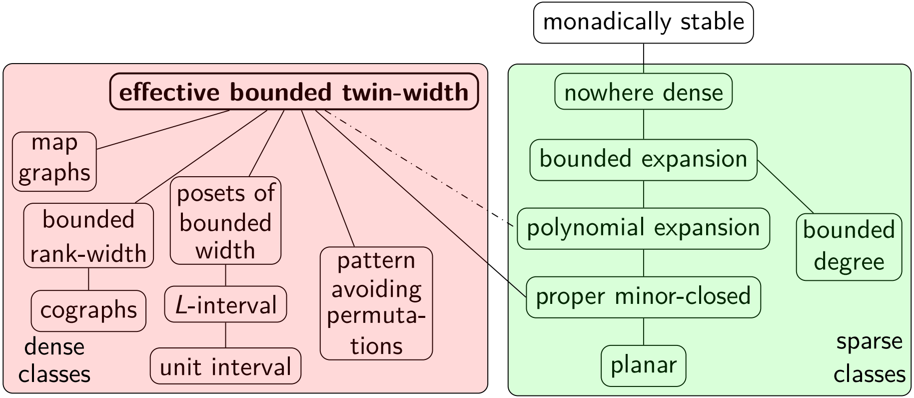
Bounded-degree graphs $\rightarrow$ stable
Unit interval graphs $\rightarrow$ dependent but not stable
Interval graphs $\rightarrow$ independent

Bounded twin-width graphs $\rightarrow$ dependent but not stable

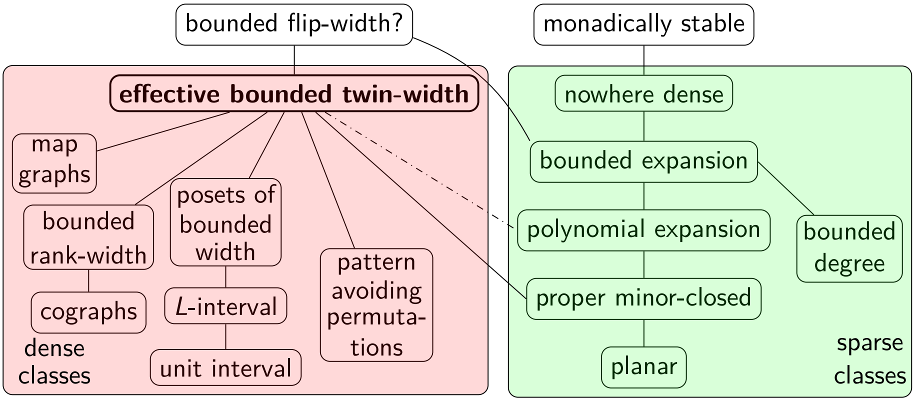
Classes for which FO model checking is FPT



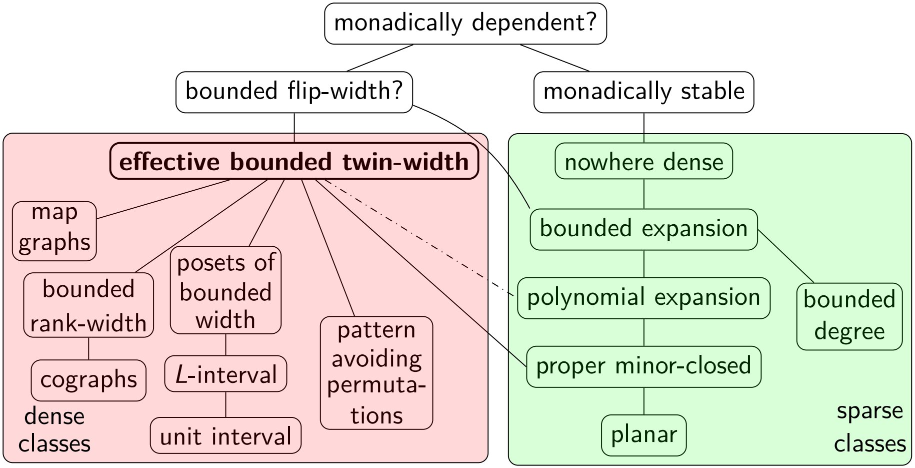
Classes for which FO model checking is FPT



Classes for which FO model checking is FPT



Classes for which FO model checking is FPT



First-order transductions and twin-width

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

For every class $\mathcal{C}$ *with bounded twin-width and transduction* T*, the class* $\mathsf{T}(\mathcal{C})$ *has bounded twin-width.*

First-order transductions and twin-width

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

For every class $\mathcal{C}$ with bounded twin-width and transduction T, the class $\mathsf{T}(\mathcal{C})$ has bounded twin-width.

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)

A class has bounded twin-width if and only if it is the transduction of a proper permutation class.

First-order transductions and twin-width

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

For every class $\mathcal{C}$ with bounded twin-width and transduction T, the class $\mathsf{T}(\mathcal{C})$ has bounded twin-width.

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)

A class has bounded twin-width if and only if it is the transduction of a proper permutation class.

Theorem (B., Bourneuf, Geniet, Thomassé '24)

There is a fixed proper permutation class $\mathcal{P}$ such that a class has bounded twin-width if and only if it is the transduction of $\mathcal{P}$.

The lens of contraction sequences

Class of bounded	FO transduction of	constr. on red graphs	efficient MC
linear rank-width	linear order	bd #edges	MSO
rank-width	tree order	bd component	**MSO**
twin-width	**proper perm. class**	bd degree	**FO**

Equivalences for **ordered** graphs

Theorem (B., Giocanti, Ossona de Mendez, Toruńczyk, Thomassé, Simon '21)

Let $\mathcal{C}$ be a hereditary class of ordered graphs. TFAE:

(*i*) *$\mathcal{C}$ has bounded twin-width.*

(*ii*) *$\mathcal{C}$ has a tractable FO model checking.*

(*iii*) *$\mathcal{C}$ is monadically dependent.*

(*iv*) *$\mathcal{C}$ has single-exponential growth.*

(*v*) *$\mathcal{C}$ has subfactorial growth.*

Equivalences for **ordered** graphs

Theorem (B., Giocanti, Ossona de Mendez, Toruńczyk, Thomassé, Simon '21)

Let $\mathcal{C}$ be a hereditary class of ordered graphs. TFAE:

(*i*) *$\mathcal{C}$ has bounded twin-width.*

(*ii*) *$\mathcal{C}$ has a tractable FO model checking.*

(*iii*) *$\mathcal{C}$ is monadically dependent.*

(*iv*) *$\mathcal{C}$ has single-exponential growth.*

(*v*) *$\mathcal{C}$ has subfactorial growth.*

Bounded twin-width is the structural characterization of tame ordered binary structures

Open questions

- Algorithm to compute/approximate twin-width in general
- Fully classify classes with tractable FO model checking
- Constructions of subcubic unbounded twin-width graphs
- Better approximations on bounded twin-width classes

Open questions

- Algorithm to compute/approximate twin-width in general
- Fully classify classes with tractable FO model checking
- Constructions of subcubic unbounded twin-width graphs
- Better approximations on bounded twin-width classes

Thank you for your attention!