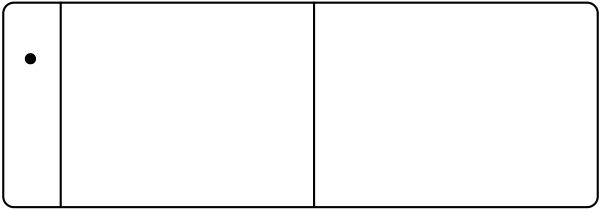
Twin-width

Édouard Bonnet
based on joint works with Colin Geniet, Ugo Giocanti, Eunjung Kim, Jarik Nešetřil, Patrice Ossona de Mendez, Amadeus Reinald, Sebastian Siebertz, Pierre Simon, Stéphan Thomassé, Szymon Toruńczyk, and Rémi Watrigant

ENS Lyon, LIP

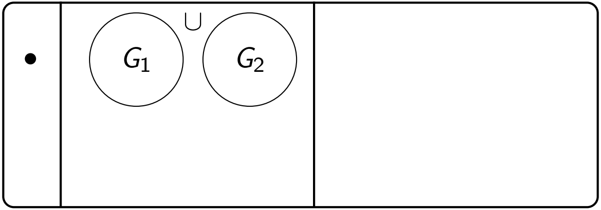
June 25th, 2021, WG

Cographs



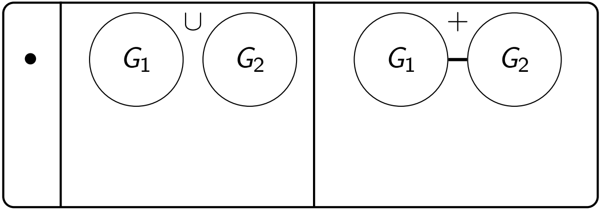
A single vertex is a cograph,

Cographs



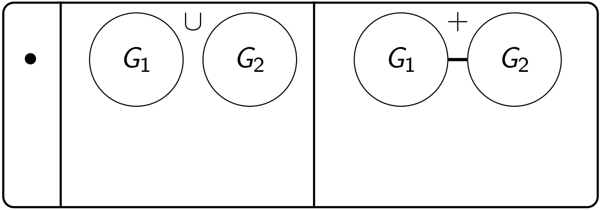
as well as the union of two cographs,

Cographs

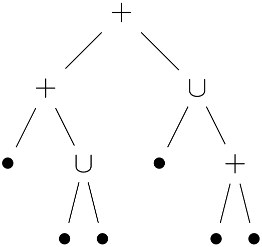


and the complete join of two cographs.

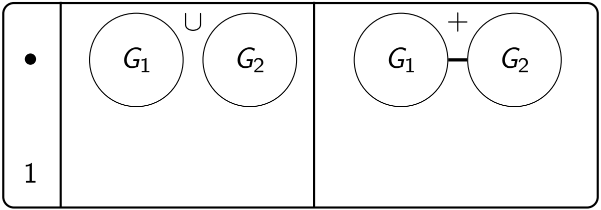
Cographs



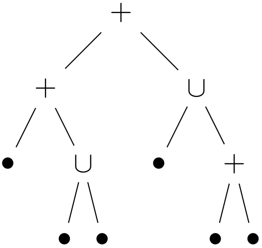
Many NP-hard problems are polytime solvable on cographs



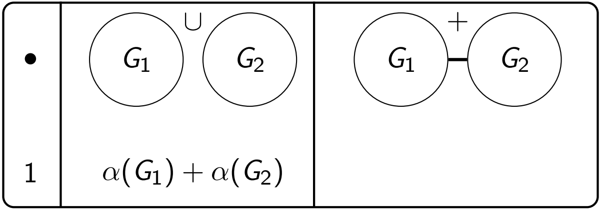
Cographs



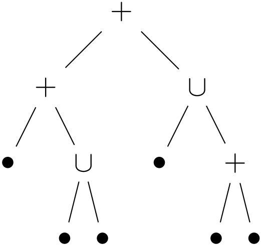
Let's try to compute the NP-hard $\alpha(G)$, independence number



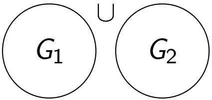
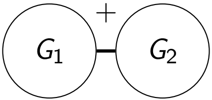
Cographs



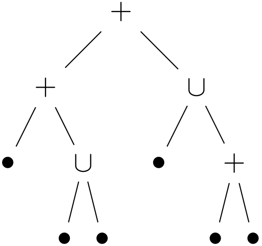
In case of a disjoint union: combine the solutions



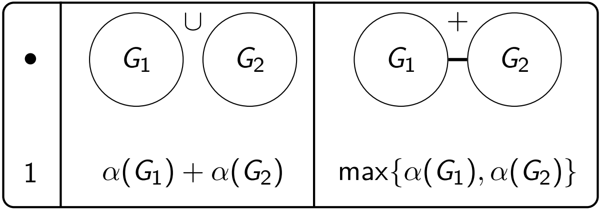
Cographs

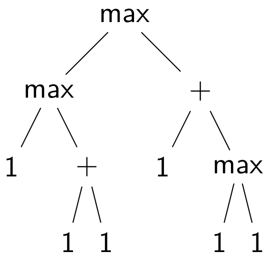
•		
1	$\alpha(G_1) + \alpha(G_2)$	$\max\{\alpha(G_1), \alpha(G_2)\}$

In case of a complete join: pick the larger one



Cographs


•
1
G_1 ∪ G_2
$\alpha(G_1) + \alpha(G_2)$
G_1 + G_2
$\max\{\alpha(G_1), \alpha(G_2)\}$


max
max
+
1
+
1
max
1 1
1 1

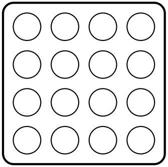
Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

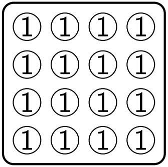


Is there another algorithmic scheme based on this definition?

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

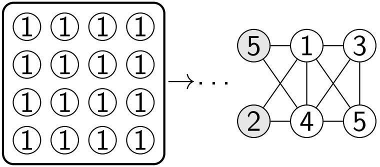


Let's try with $\alpha(G)$, and store in a vertex its inner max solution

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

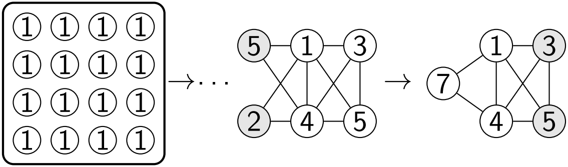


We can find a pair of false/true twins

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

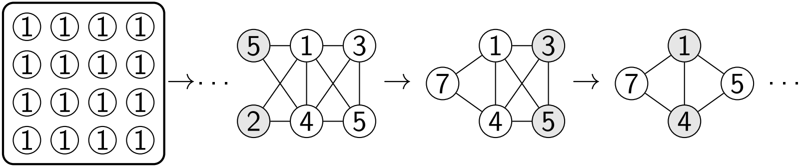


Sum them if they are false twins

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

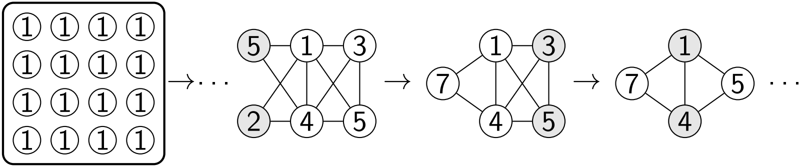


Max them if they are true twins

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

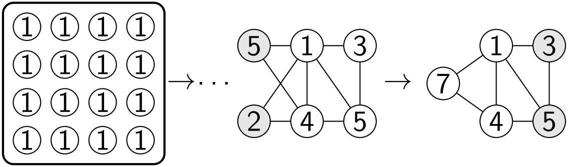


The eventual single vertex contains $\alpha(G)$

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

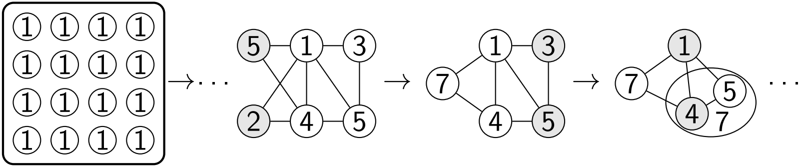


What if we only have near twins?

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

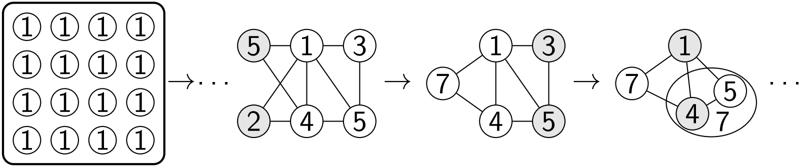


A few partial solutions to memorize

[1] provided it has at least two vertices

Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every[1] graph has two *twins*

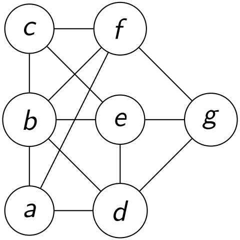


A few partial solutions to memorize

Errors occur in bounded size components → bounded rank-width
Errors occur with bounded degree → bounded twin-width

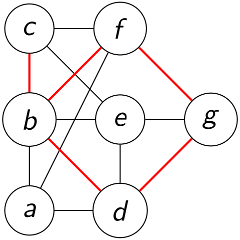
[1] provided it has at least two vertices

Graphs



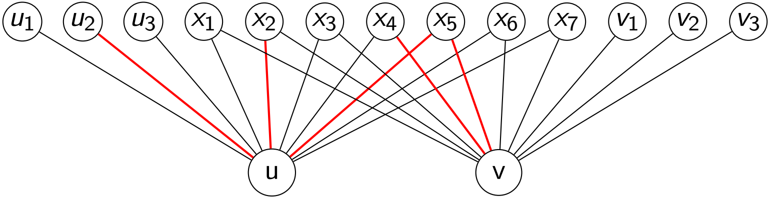
Two outcomes between a pair of vertices:
edge or non-edge

Trigraphs



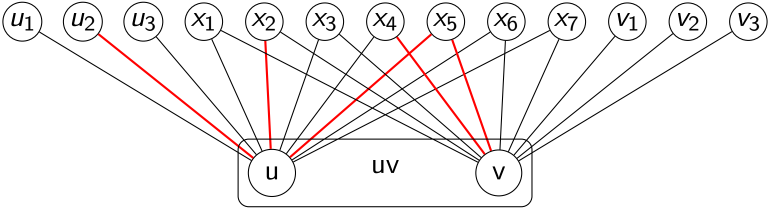
Three outcomes between a pair of vertices:
edge, or non-edge, or red edge (error edge)

Contractions in trigraphs



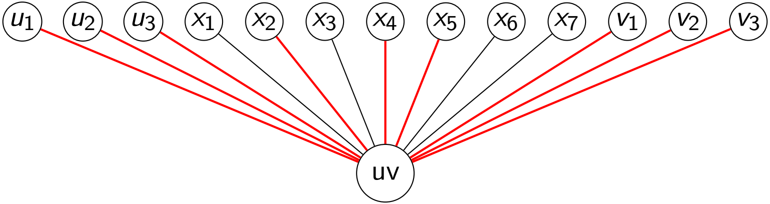
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



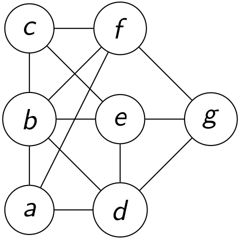
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



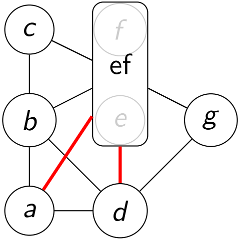
edges to $N(u) \triangle N(v)$ turn red, for $N(u) \cap N(v)$ red is absorbing

Contraction sequence



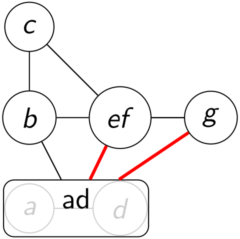
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



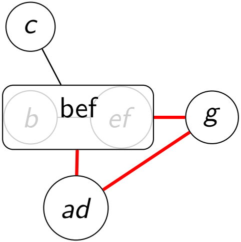
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



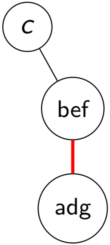
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \dots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



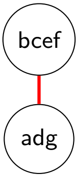
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



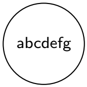
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

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}.

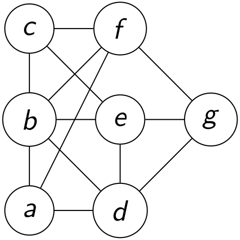
Contraction sequence



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}.

Twin-width

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

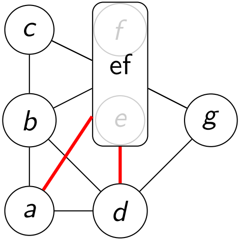


Maximum red degree $= 0$

overall maximum red degree = 0

Twin-width

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

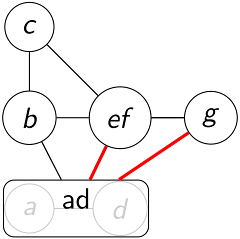


Maximum red degree $= 2$

overall maximum red degree = 2

Twin-width

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

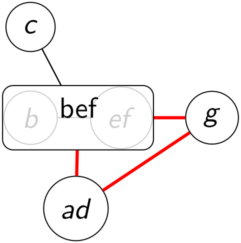


Maximum red degree $= 2$

overall maximum red degree = 2

Twin-width

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

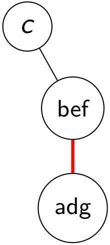


Maximum red degree = 2

overall maximum red degree = 2

Twin-width

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

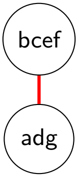


Maximum red degree $= 1$

overall maximum red degree = 2

Twin-width

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

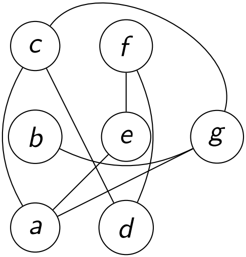
Maximum red degree $= 0$

overall maximum red degree = 2

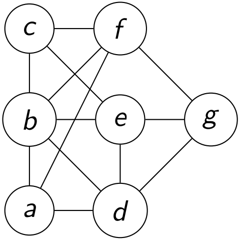
Simple operations preserving small twin-width

- **complementation: remains the same**
- taking induced subgraphs: may only decrease
- adding one vertex linked arbitrarily: at most "doubles"
- **modular decomposition: max of the twin-widths**

Complementation



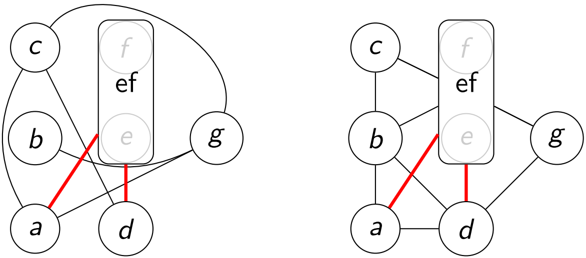
$\overline{G}$



G

$$\text{tww}(\overline{G}) = \text{tww}(G)$$

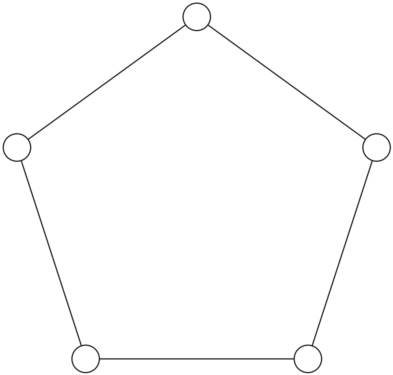
Complementation



$\overline{G_6}$ $\quad$ G_6

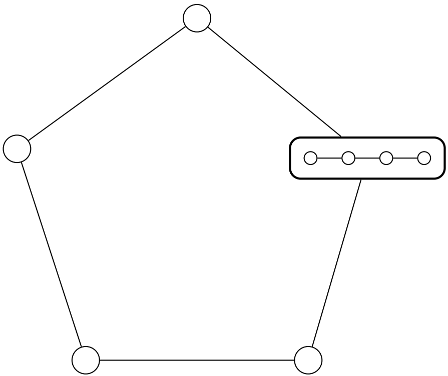
$$\text{tww}(\overline{G}) = \text{tww}(G)$$

Substitution and lexicographic product



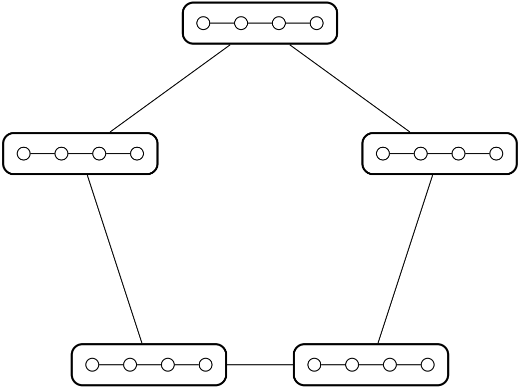
$G = C_5$

Substitution and lexicographic product



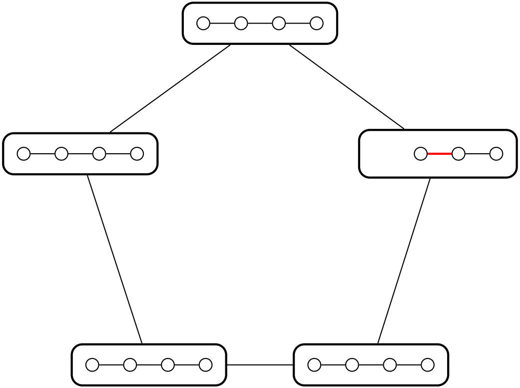
$G = C_5$, $H = P_4$, substitution $G[v \leftarrow H]$

Substitution and lexicographic product



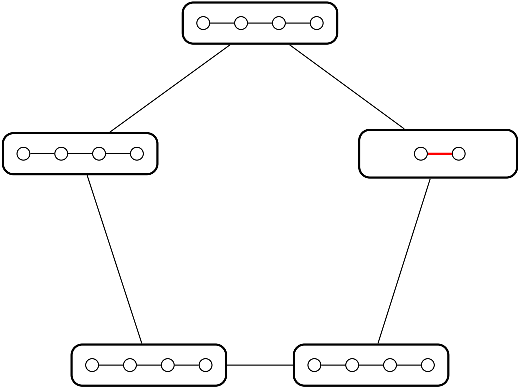
$G = C_5$, $H = P_4$, lexicographic product $G[H]$

Substitution and lexicographic product



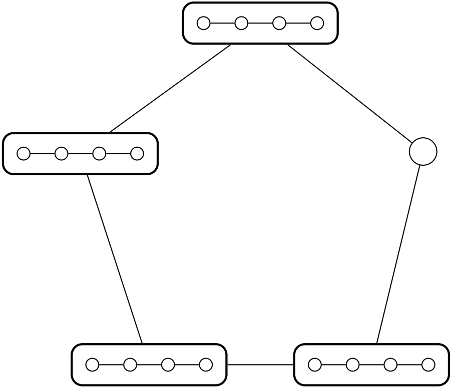
More generally any modular decomposition

Substitution and lexicographic product



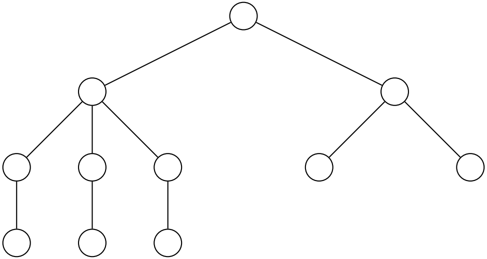
More generally any modular decomposition

Substitution and lexicographic product



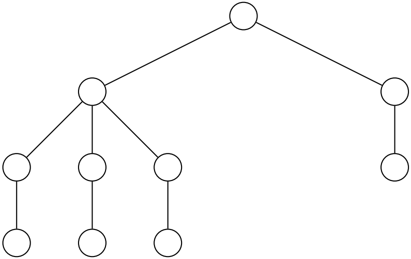
$$\text{tww}(G[H]) = \max(\text{tww}(G), \text{tww}(H))$$

Graphs with bounded twin-width – trees



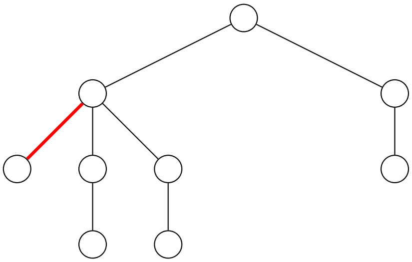
If possible, contract two twin leaves

Graphs with bounded twin-width – trees



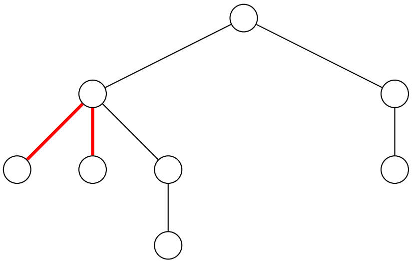
If not, contract a deepest leaf with its parent

Graphs with bounded twin-width – trees



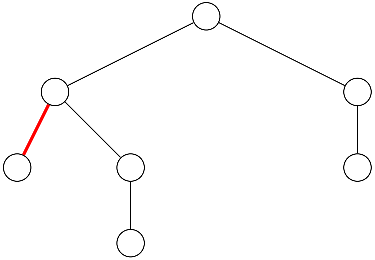
If not, contract a deepest leaf with its parent

Graphs with bounded twin-width – trees



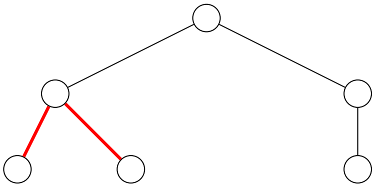
If possible, contract two twin leaves

Graphs with bounded twin-width – trees



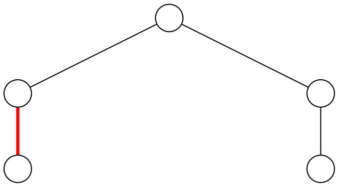
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



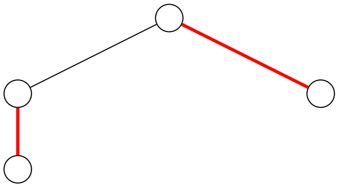
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



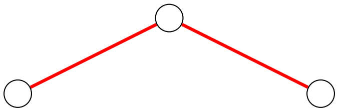
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



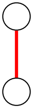
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees

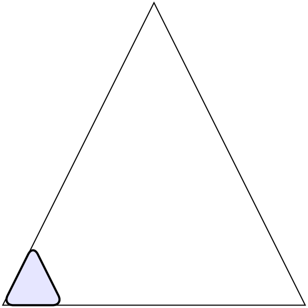


Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees

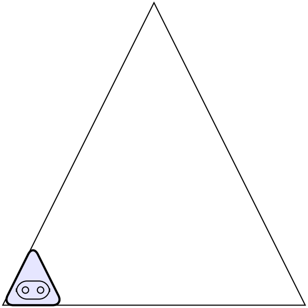
Cannot create a red degree-3 vertex

Graphs with bounded twin-width – trees



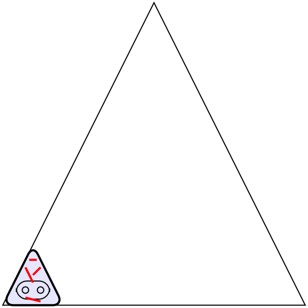
Generalization to bounded *rank-width*

Graphs with bounded twin-width – trees



Two near-twins in a small subtree → contraction

Graphs with bounded twin-width – trees

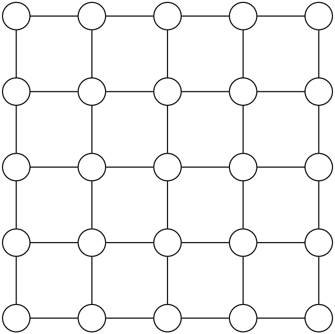


Red edges cluster in bounded size components

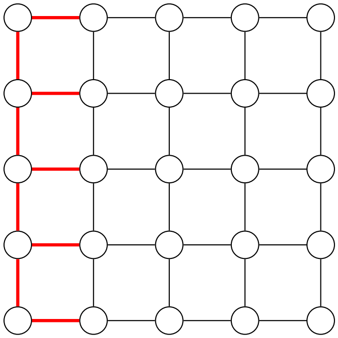
Theorem (B., Kim, Reinald, Thomassé, Watrigant, '21+)

A class has bounded rank-width if and only if its graphs admit contraction sequences for which the red components are bounded.

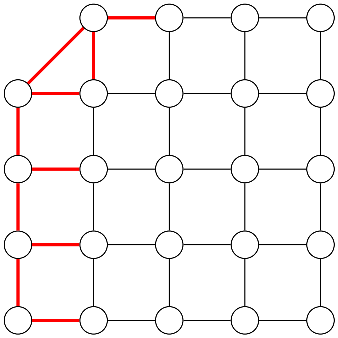
Graphs with bounded twin-width – grids



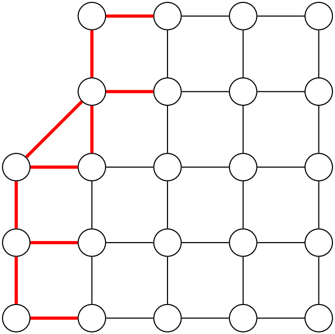
Graphs with bounded twin-width – grids



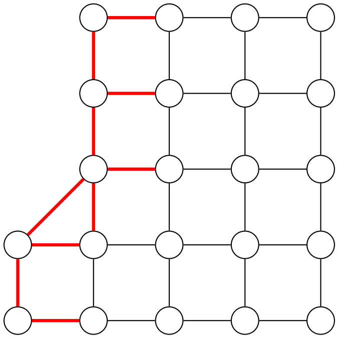
Graphs with bounded twin-width – grids



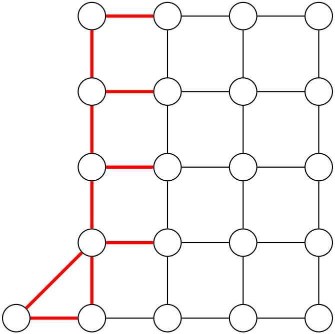
Graphs with bounded twin-width – grids



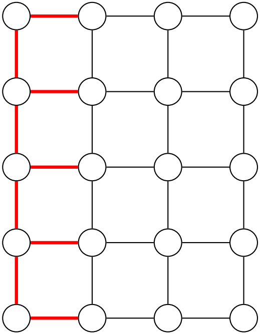
Graphs with bounded twin-width – grids



Graphs with bounded twin-width – grids



Graphs with bounded twin-width – grids



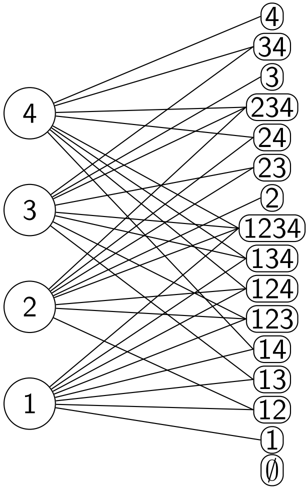
4-sequence for planar grids, $3d$-sequence for d-dimensional grids

Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.

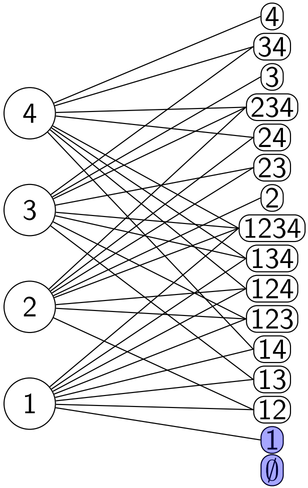
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



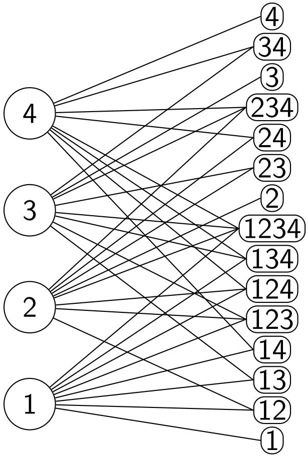
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



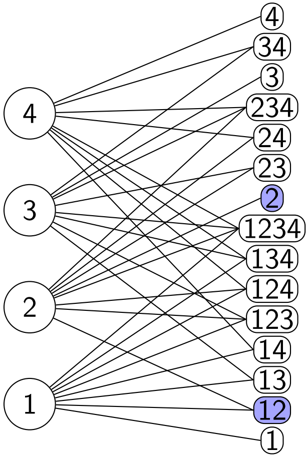
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



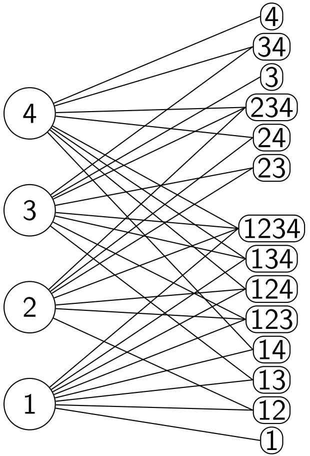
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



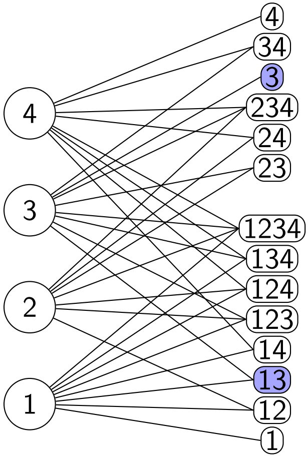
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



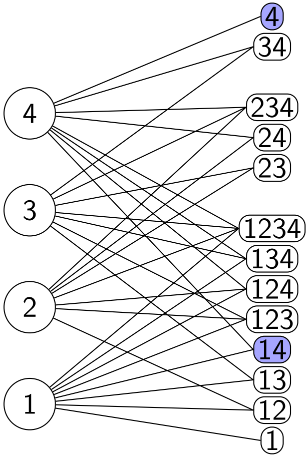
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



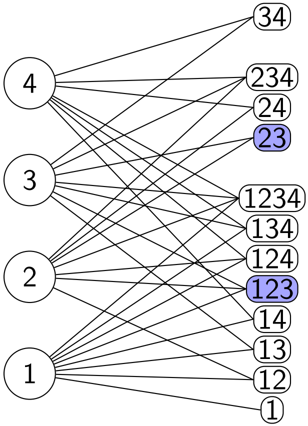
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



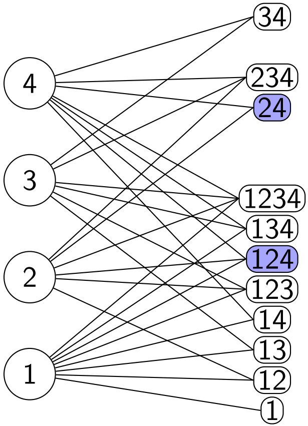
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



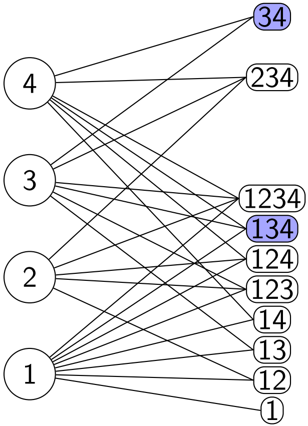
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



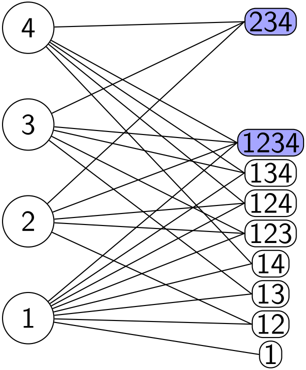
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



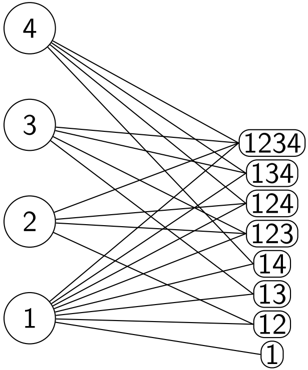
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



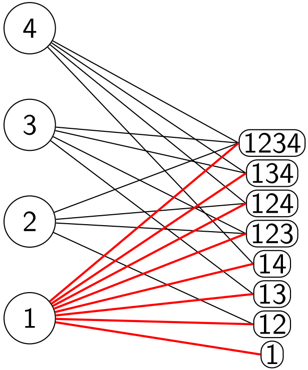
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



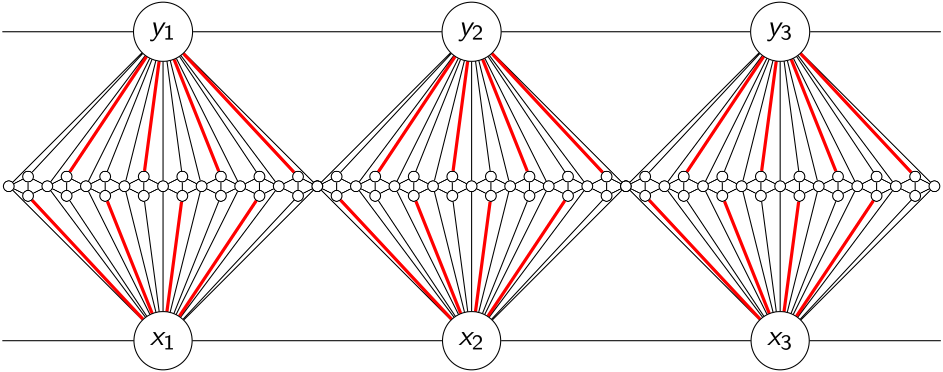
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.



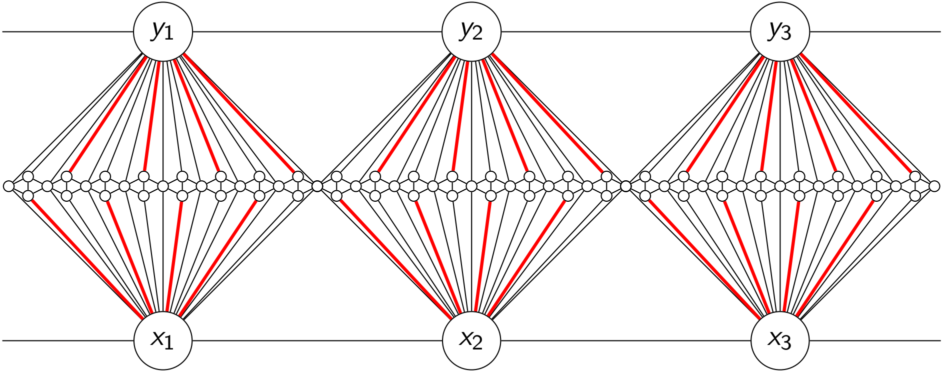
Graphs with bounded twin-width – planar graphs?

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

Twin-width in the language of matrices

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Encode a bipartite graph (or, if symmetric, any graph)

Twin-width in the language of matrices

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Contraction of two columns (similar with two rows)

Twin-width in the language of matrices

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & \color{red}{r} & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & \color{red}{r} & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & \color{red}{r} & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

How is the twin-width (re)defined?

Twin-width in the language of matrices

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & r & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & r & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & r & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

How to tune it for non-bipartite graph?

Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are *consecutive*

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

Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are *consecutive*

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Maximum number of non-constant zones per column or row part
= **error value**

Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are *consecutive*

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

Maximum number of non-constant zones per column or row part
. . . until there are a single row part and column part

Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are *consecutive*

$$
\begin{bmatrix}
1 & 1 & 1 \;\; 1 & 1 & 1 & 1 & 0 \\ \hline
0 & 1 & 1 \;\; 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 \;\; 0 & 0 & 0 & 0 & 1 \\ \hline
0 & 1 & 0 \;\; 0 & 1 & 0 & 1 & 0 \\ \hline
1 & 0 & 0 \;\; 1 & 1 & 0 & 1 & 0 \\ \hline
0 & 1 & 1 \;\; 1 & 1 & 1 & 0 & 0 \\ \hline
1 & 0 & 1 \;\; 1 & 1 & 0 & 0 & 1
\end{bmatrix}
$$

Twin-width as maximum error value of a contraction sequence

Grid minor

t-grid minor: $t \times t$-division where every cell is non-empty
Non-empty cell: contains at least one 1 entry

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

4-grid minor

Grid minor

t-grid minor: $t \times t$-division where every cell is non-empty
Non-empty cell: contains at least one 1 entry

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

4-grid minor

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

Mixed minor

Mixed cell: 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

Mixed minor

Mixed cell: 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

Every mixed cell is witnessed by a 2×2 square = **corner**

Mixed minor

Mixed cell: 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

Twin-width and mixed freeness

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

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

Twin-width and mixed freeness

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

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

Twin-width and mixed freeness

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

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

Step 1: find a division sequence $(\mathcal{D}_i)_i$ **with mixed value** $f(t)$

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Merge consecutive parts greedily

Twin-width and mixed freeness

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

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

Step 1: find a division sequence $(\mathcal{D}_i)_i$ **with mixed value** $f(t)$

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Merge consecutive parts greedily

Twin-width and mixed freeness

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

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

Step 1: find a division sequence $(\mathcal{D}_i)_i$ with mixed value $f(t)$

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Merge consecutive parts greedily

Twin-width and mixed freeness

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

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

Step 1: find a division sequence $(\mathcal{D}_i)_i$ **with mixed value** $f(t)$

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Stuck, removing every other separation $\rightarrow \frac{f(t)}{2}$ mixed cells per part

Stanley-Wilf conjecture/Marcus-Tardos theorem

Theorem (Marcus, Tardos '04)

For every k, there is a c_k such that every $n \times m$ $0,1$-matrix with at least $c_k \max(n, m)$ 1 entries admits a k-grid minor.

Twin-width and mixed freeness

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

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

Step 1: find a division sequence $(\mathcal{D}_i)_i$ **with mixed value** $f(t)$

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Stuck, removing every other separation $\rightarrow \frac{f(t)}{2}$ mixed cells per part

Impossible!

Twin-width and mixed freeness

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

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

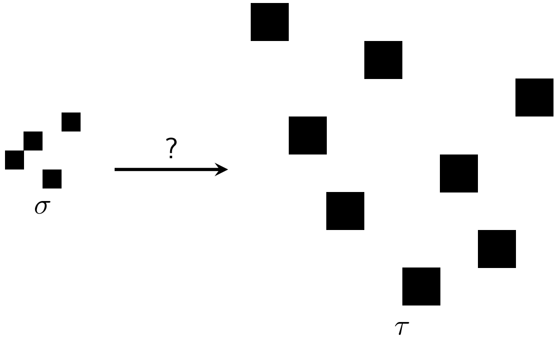
Step 1: find a division sequence $(\mathcal{D}_i)_i$ **with mixed value** $f(t)$
Step 2: find a contraction sequence with error value $g(t)$

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

Refinement of $\mathcal{D}_i$ where each part coincides on the non-mixed cells

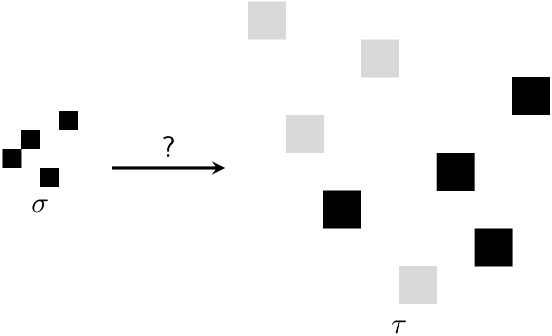
Follows Guillemot & Marx algorithm '14

Linear FPT algorithm for PERMUTATION PATTERN



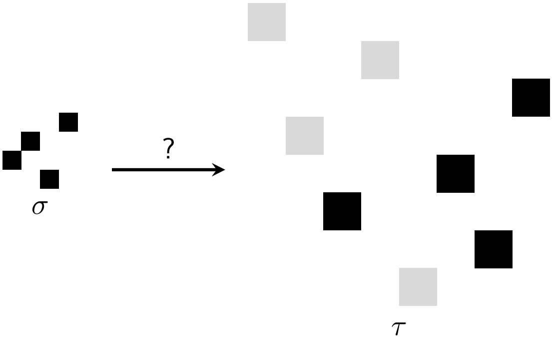
Follows Guillemot & Marx algorithm '14

Linear FPT algorithm for PERMUTATION PATTERN



Follows Guillemot & Marx algorithm '14

Linear FPT algorithm for PERMUTATION PATTERN



"It would be interesting to see if there is a corresponding graph-theoretic analog for this scheme, which might be useful for solving some graph-theoretical problem."

Twin-width and mixed freeness

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

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

Twin-width and mixed freeness

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

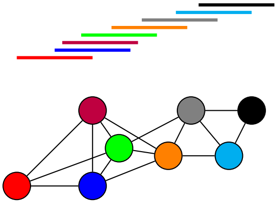
If $\exists \sigma$ s.t. $Adj_\sigma(G)$ is t-mixed free, then $tww(G) = 2^{2^{O(t)}}$.

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

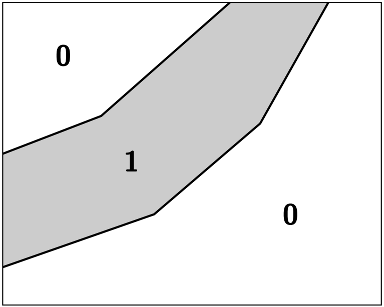
1) Find a *good* vertex-ordering procedure
2) Argue that, in this order, a t-mixed minor would conflict with $\mathcal{C}$

Unit interval graphs

Intersection graph of unit segments on the real line

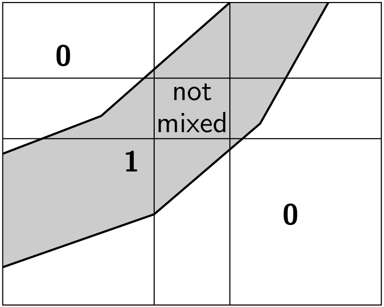


Bounded twin-width – unit interval graphs



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

Graph minors

Formed by **vertex deletion**, **edge deletion**, and **edge contraction**

A graph G is *H-minor free* if H is not a minor of G

A graph class is *H-minor free* if all its graphs are

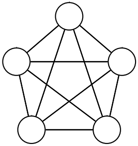
Graph minors

Formed by **vertex deletion**, **edge deletion**, and **edge contraction**

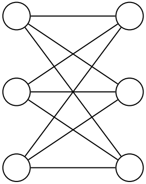
A graph G is *H-minor free* if H is not a minor of G

A graph class is *H-minor free* if all its graphs are

Planar graphs are exactly the graphs without K_5 or $K_{3,3}$ as a minor



K_5

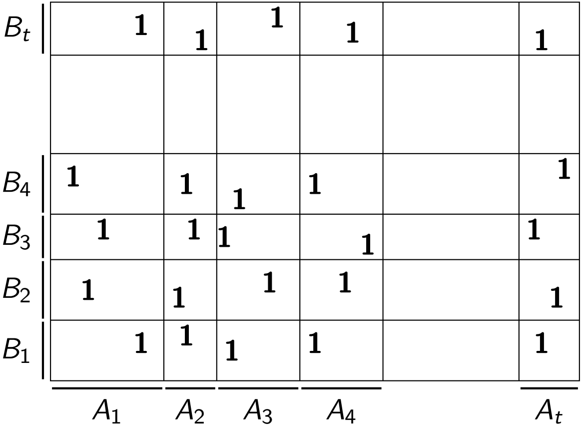


$K_{3,3}$

Bounded twin-width – K_t-minor free graphs

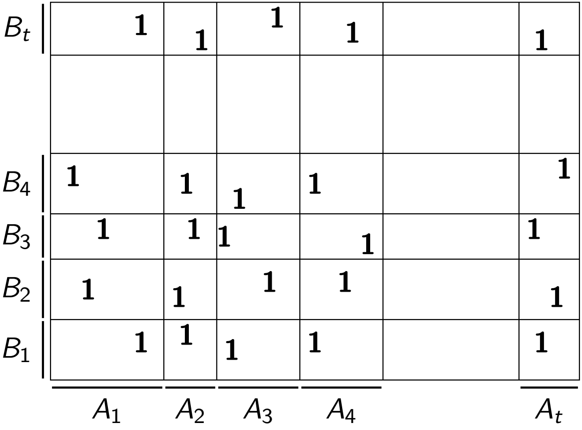
Given a hamiltonian path, we would just use this order

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

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

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

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.

Example of k-Independent Set

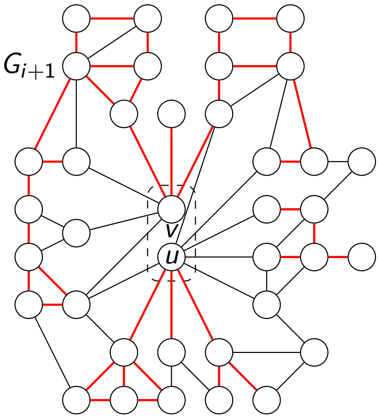
d-sequence: $G = G_n, G_{n-1}, \dots, G_2, G_1 = K_1$

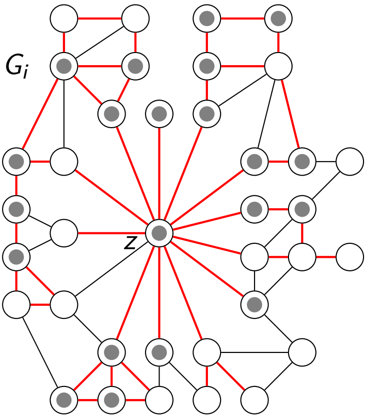
Algorithm: **Compute by dynamic programming a best partial solution in each red connected subgraph of size at most k.**

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

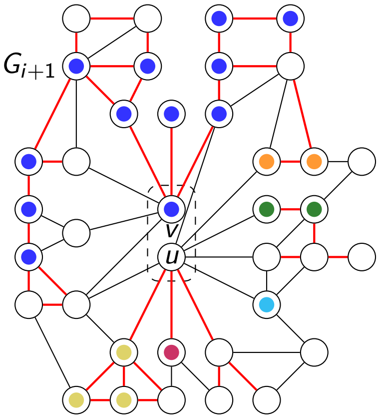
In G_n: red connected subgraphs are singletons, so are the solutions.
In G_1: If solution of size at least k, global solution.

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

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

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:** $\vert\varphi\vert$
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:** $|\varphi|$
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:** $|\varphi|$
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 simple 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)

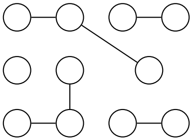
FO interpretations and transductions

FO simple 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



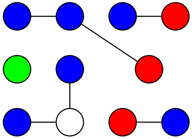
FO interpretations and transductions

FO simple 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



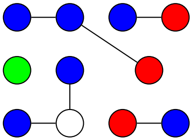
FO interpretations and transductions

FO simple 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



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

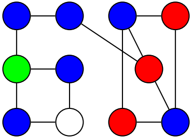
FO interpretations and transductions

FO simple 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



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

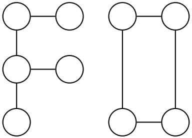
FO interpretations and transductions

FO simple 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



Dependence and monadic dependence

A class $\mathcal{C}$ is
dependent, if the hereditary closure of every interpretation of $\mathcal{C}$ misses some graph
monadically dependent, if every transduction of $\mathcal{C}$ misses some graph [Baldwin, Shelah '85]

Dependence and monadic dependence

A class $\mathcal{C}$ is
dependent, if the hereditary closure of every interpretation of $\mathcal{C}$ misses some graph
monadically dependent, if every transduction of $\mathcal{C}$ misses some graph [Baldwin, Shelah '85]

Theorem (Downey, Fellows, Taylor '96)

FO model checking is $AW[*]$-complete on general graphs,

Thus unlikely tractable on *effectively* independent class

Dependence and monadic dependence

A class $\mathcal{C}$ is
dependent, if the hereditary closure of every interpretation of $\mathcal{C}$ misses some graph
monadically dependent, if every transduction of $\mathcal{C}$ misses some graph [Baldwin, Shelah '85]

Theorem (Downey, Fellows, Taylor '96)

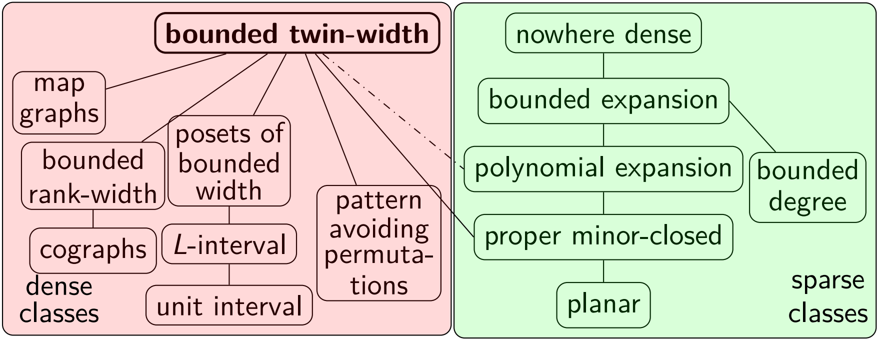
FO model checking is $AW[*]$-complete on general graphs,

Thus unlikely tractable on *effectively* independent class

Conjecture (Gajarský et al. '18)

Every dependent class is tractable.

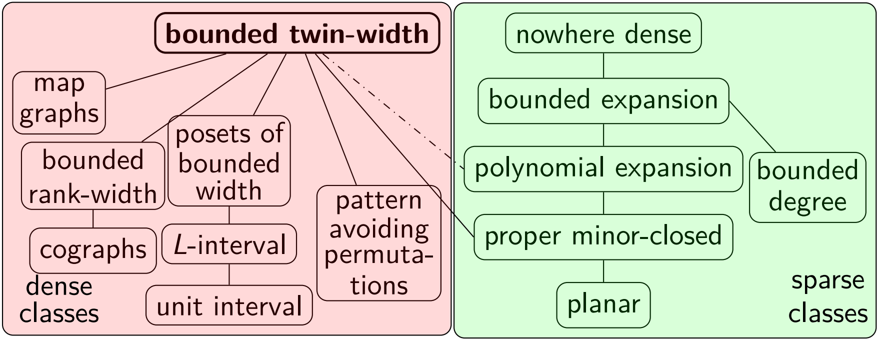
Known FPT FO model checking



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

FO MODEL CHECKING *solvable in* $f(|\varphi|, d)n$ *on graphs with a d-sequence.*

Known FPT FO model checking



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

FO MODEL CHECKING *solvable in* $f(|\varphi|, d)n$ *on graphs with a* d*-sequence.*

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

Any transduction of a bounded twin-width class has bounded twin-width.

Small classes

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

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

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 '21)

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 '21)

Bounded twin-width classes are small.

Is the converse true for hereditary classes?

Conjecture (small conjecture, refuted)

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

Equivalences for **ordered** graphs

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

Let $\mathcal{C}$ be a hereditary class of ordered graphs, the following are equivalent.

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

(*ii*) $\mathcal{C}$ *is tractable.*

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

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

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

(*vi*) $\mathcal{C}$ *is small.*

Equivalences for **ordered** graphs

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

Let $\mathcal{C}$ be a hereditary class of ordered graphs, the following are equivalent.

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

(*ii*) *$\mathcal{C}$ is tractable.*

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

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

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

(*vi*) *$\mathcal{C}$ is small.*

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

Model-theoretic characterizations

A class of binary structures has bounded **twin-width** if and only if

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

. . . it is an FO transduction of a ***proper permutation class****.*

rank-width $\cdots$ tree order
linear rank-width $\cdots$ linear order

Model-theoretic characterizations

A class of binary structures has bounded **twin-width** if and only if

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

*. . . it is an FO transduction of a **proper permutation class**.*

Theorem (Tww I + Tww IV)

. . . it is the reduct of a monadically dependent class of ordered binary structures.

χ-boundedness

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

Theorem

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

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.

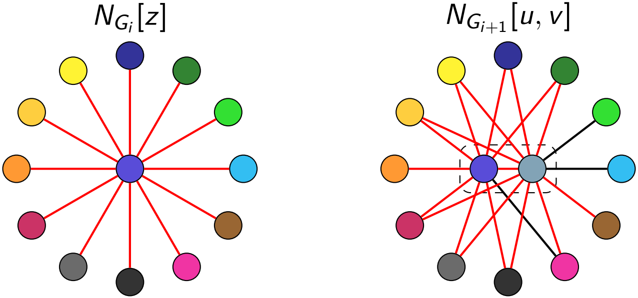
Polynomially χ-bounded? i.e., $\chi(G) = O(\omega(G)^d)$

$d + 2$-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

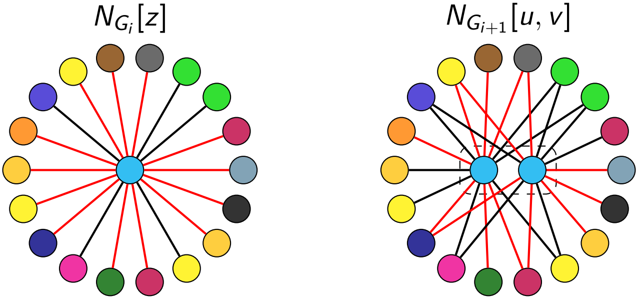
Algorithm: **Start from $G_1 = K_1$, color its unique vertex 1, and rewind the d-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.**



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

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
- Polynomial χ-boundedness of bounded twin-width classes
- Faster linear algebra on bounded twin-width matrices

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
- Polynomial χ-boundedness of bounded twin-width classes
- Faster linear algebra on bounded twin-width matrices

Thank you for your attention!