A gentle introduction to twin-width

Édouard Bonnet
based on joint works with Colin Geniet, Eunjung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant

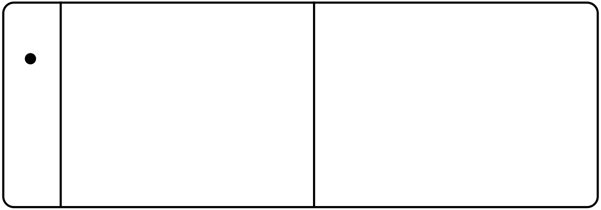
ENS Lyon, LIP

January 10th, 2023, Computer Science Seminar, Liverpool

Profession of faith in algorithmic graph theory

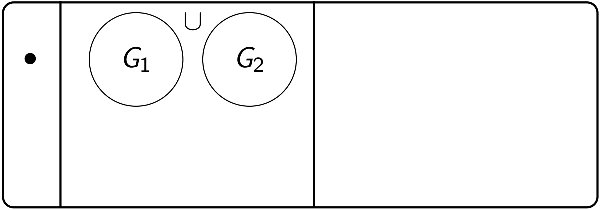
- General graphs are tough
- Real-life networks are structured
- Let us try to exploit that structure

Cographs



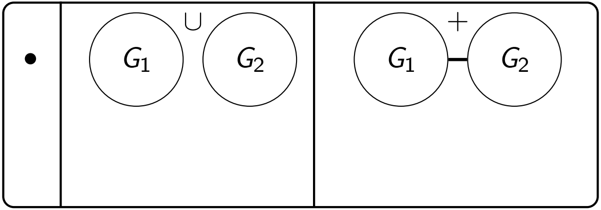
A single vertex is a cograph,

Cographs



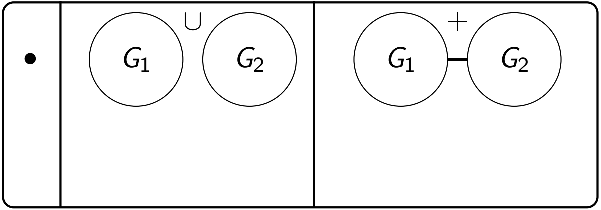
as well as the union of two cographs,

Cographs

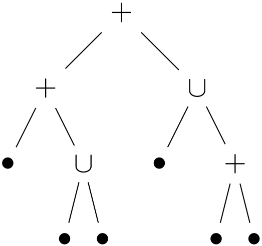


and the complete join of two cographs.

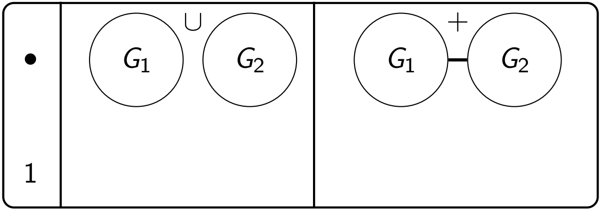
Cographs



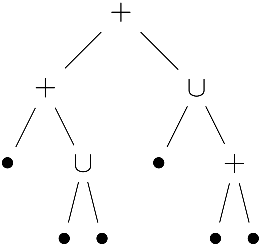
Many NP-hard problems are polytime solvable on cographs



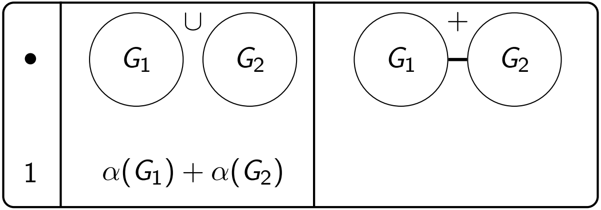
Cographs



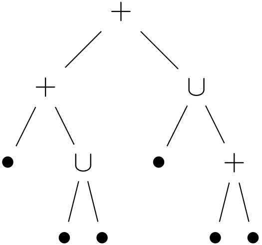
For instance the independence number $\alpha(G)$ is polytime



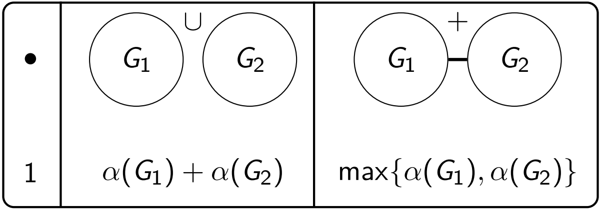
Cographs



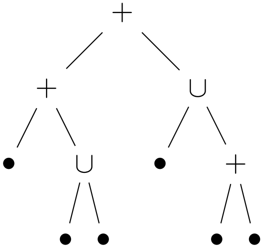
In case of a disjoint union: combine the solutions



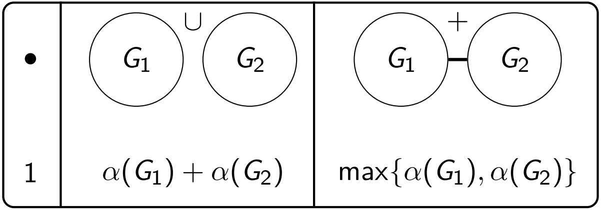
Cographs

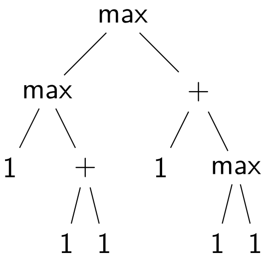


In case of a complete join: pick the larger one



Cographs


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

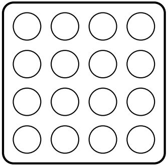

max
max
+
1
+
1
max
1 1
1 1

Another cograph definition

Every induced subgraph has two twins

Another cograph definition

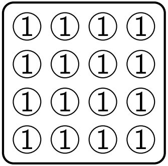
Every induced subgraph has two twins



Is there another algorithmic scheme based on this definition?

Another cograph definition

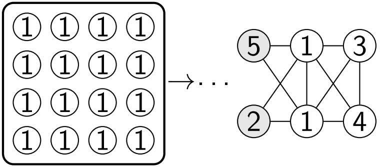
Every induced subgraph has two twins



We store in each vertex its inner max independent set

Another cograph definition

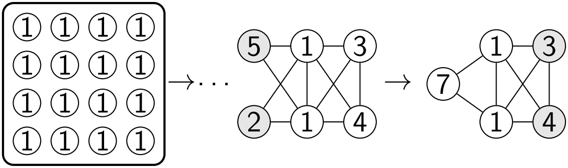
Every induced subgraph has two twins



We can find a pair of false/true twins

Another cograph definition

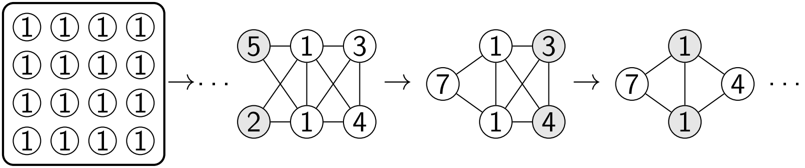
Every induced subgraph has two twins



Sum them if they are false twins

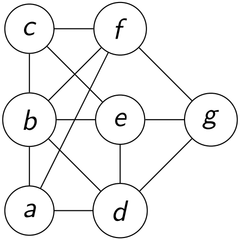
Another cograph definition

Every induced subgraph has two twins



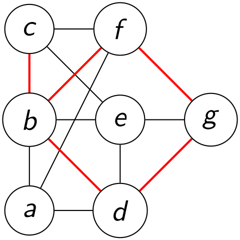
Max them if they are true twins

Generalizing the second cograph definition: going from graphs...



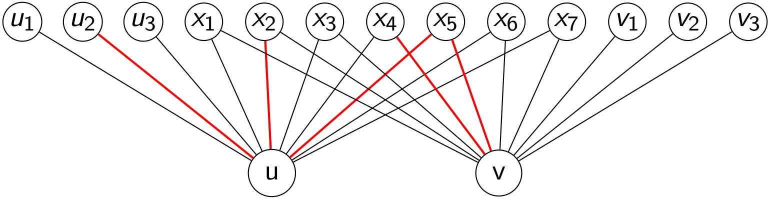
Two outcomes between a pair of vertices:
edge or non-edge

...to trigraphs



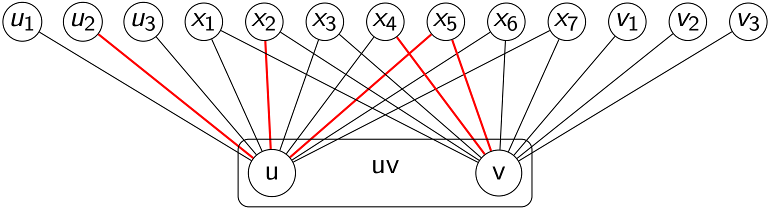
Three outcomes between a pair of vertices:
edge, or non-edge, or red edge (error edge)

Contractions in trigraphs



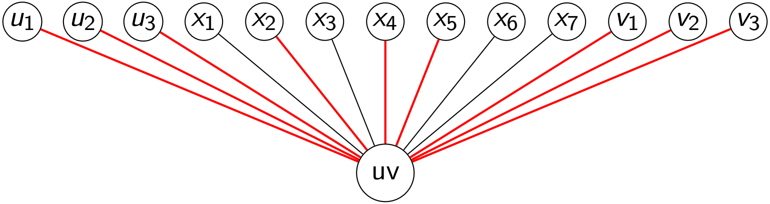
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



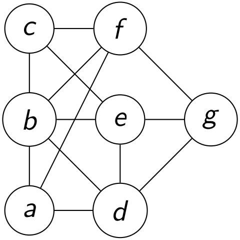
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



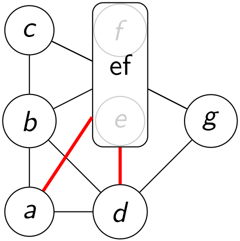
edges to $N(u) \triangle N(v)$ turn red, for $N(u) \cap N(v)$ red is absorbing

Contraction sequence



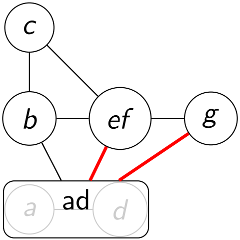
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



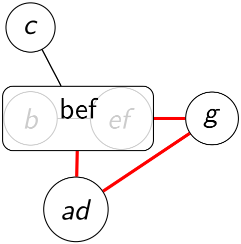
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \dots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



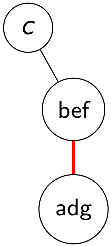
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



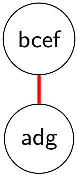
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

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



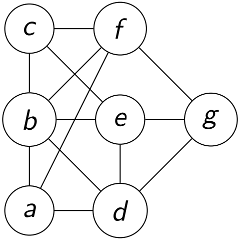
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

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}.

Twin-width

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

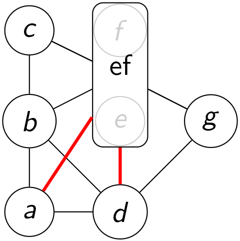


Maximum red degree = 0

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.

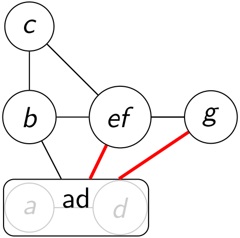


Maximum red degree $= 2$

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.

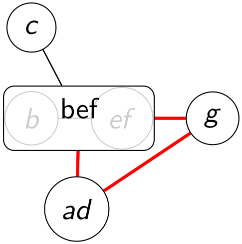


Maximum red degree $= 2$

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.

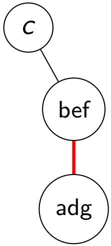


Maximum red degree = 2

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.

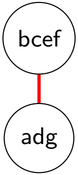


Maximum red degree $= 1$

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.

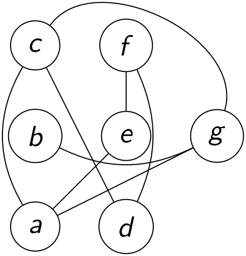
Maximum red degree = 0

overall maximum red degree = 2

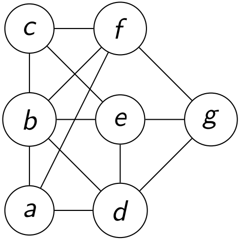
Simple operations preserving small twin-width

- complementation: remains the same
- taking induced subgraphs: may only decrease
- adding one vertex linked arbitrarily: at most "doubles"
- substitution, lexicographic product: max of the twin-widths

Complementation



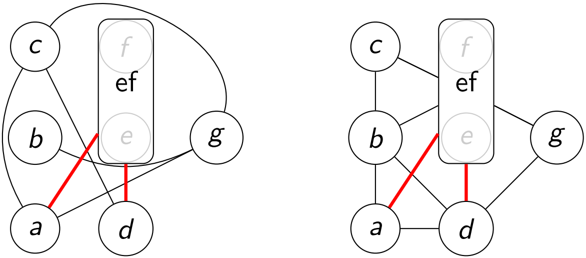
$\overline{G}$



G

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

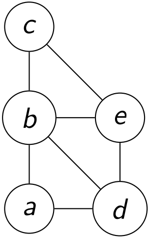
Complementation



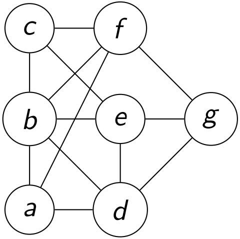
$\overline{G_6}$ G_6

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

Induced subgraph



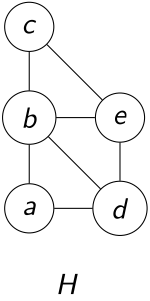
H

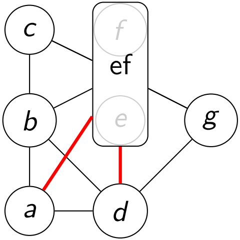


G

$$\text{tww}(H) \leqslant \text{tww}(G)$$

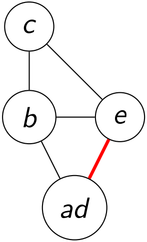
Induced subgraph

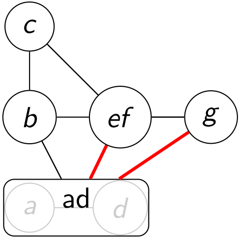




Ignore absent vertices

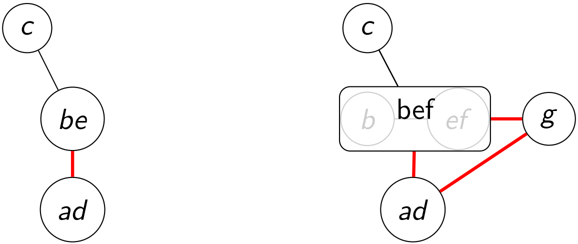
Induced subgraph





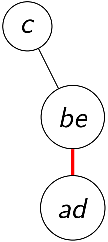
Mimic the contractions otherwise

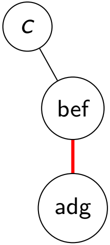
Induced subgraph



Mimic the contractions otherwise

Induced subgraph





Mimic the contractions otherwise

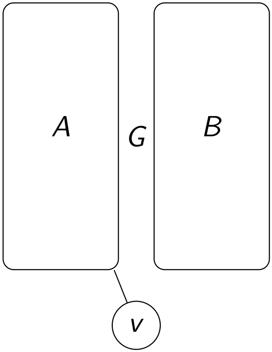
Induced subgraph

Mimic the contractions otherwise

Induced subgraph

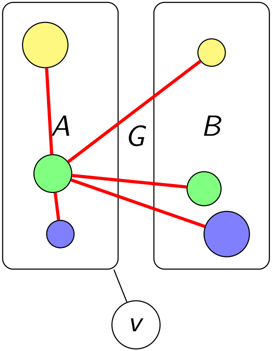
Mimic the contractions otherwise

Adding one vertex v (arbitrarily linked)



Split every part into their part in A and in B until the very end

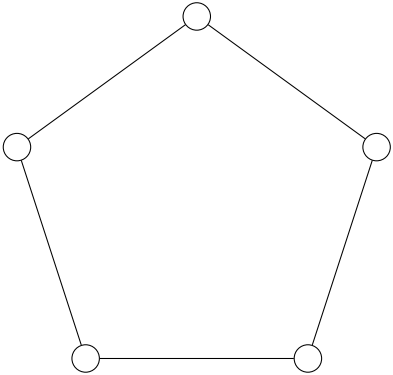
Adding one vertex v (arbitrarily linked)



Split every part into their part in A and in B until the very end

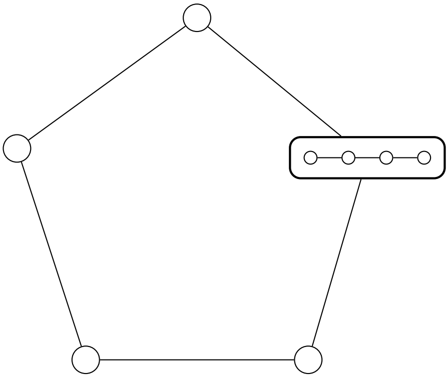
$$\mathrm{tww}(G+v) \leqslant 2 \cdot \mathrm{tww}(G) + 1$$

Substitution and lexicographic product



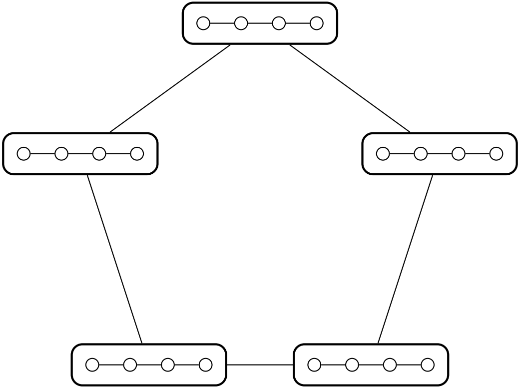
$G = C_5$

Substitution and lexicographic product



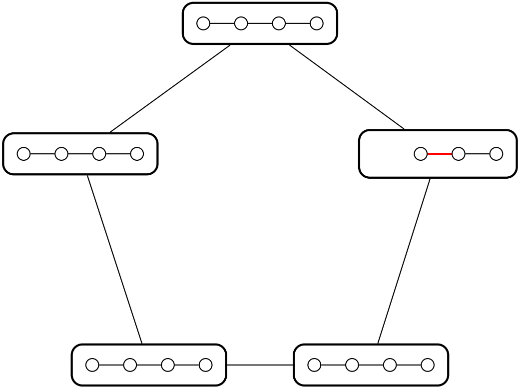
$G = C_5$, $H = P_4$, substitution $G[v \leftarrow H]$

Substitution and lexicographic product



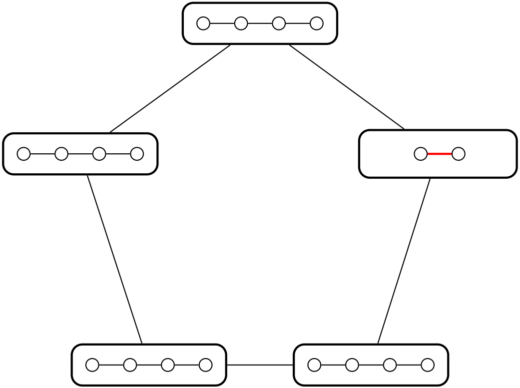
$G = C_5$, $H = P_4$, lexicographic product $G[H]$

Substitution and lexicographic product



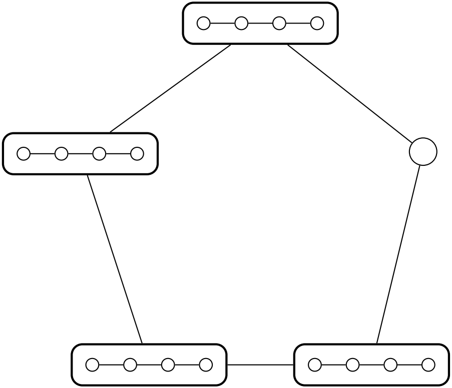
More generally any modular decomposition

Substitution and lexicographic product



More generally any modular decomposition

Substitution and lexicographic product

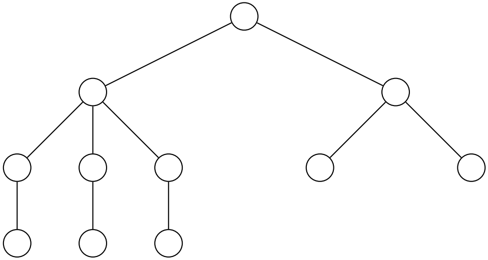


$$\mathrm{tww}(G[H]) = \max(\mathrm{tww}(G), \mathrm{tww}(H))$$

Classes with bounded twin-width

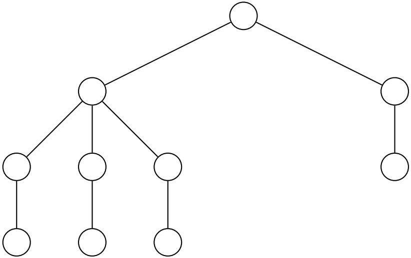
- cographs = twin-width 0
- trees
- grids
- ...

Trees



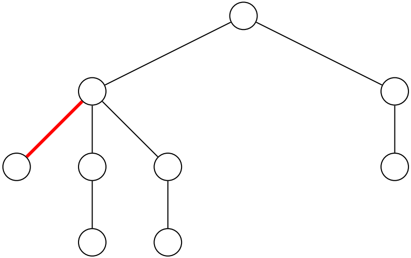
If possible, contract two twin leaves

Trees



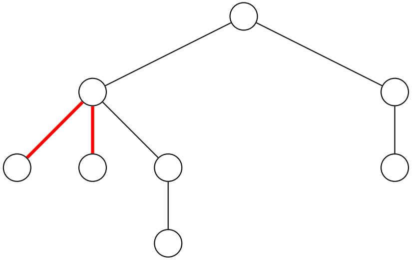
If not, contract a deepest leaf with its parent

Trees



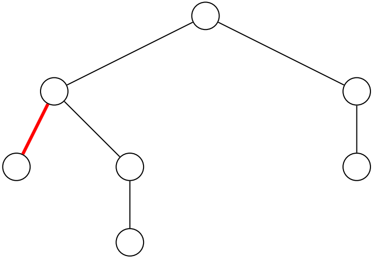
If not, contract a deepest leaf with its parent

Trees



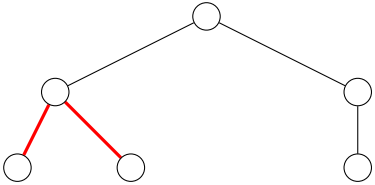
If possible, contract two twin leaves

Trees



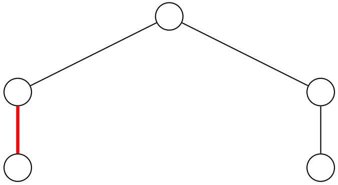
Cannot create a red degree-3 vertex

Trees



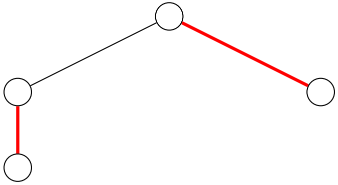
Cannot create a red degree-3 vertex

Trees



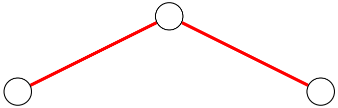
Cannot create a red degree-3 vertex

Trees



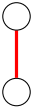
Cannot create a red degree-3 vertex

Trees



Cannot create a red degree-3 vertex

Trees

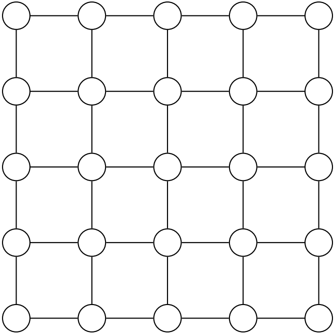


Cannot create a red degree-3 vertex

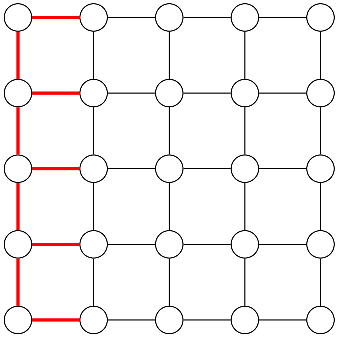
Trees

Cannot create a red degree-3 vertex

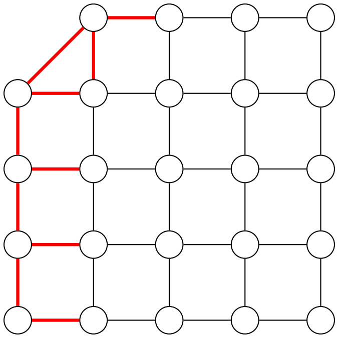
Grids



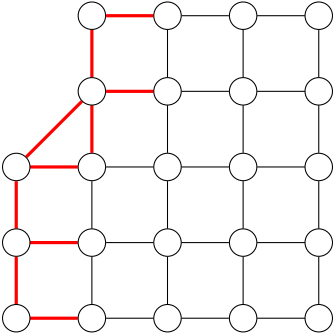
Grids



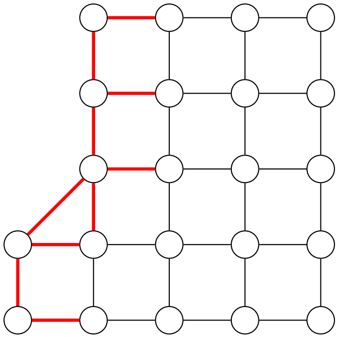
Grids



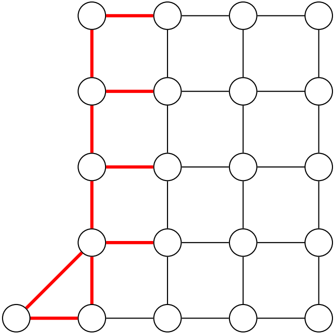
Grids



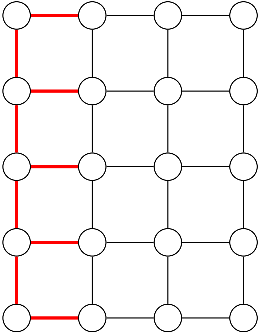
Grids



Grids

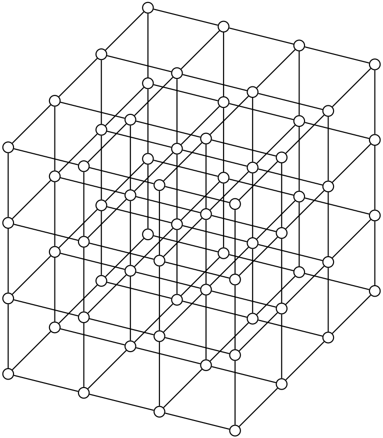


Grids



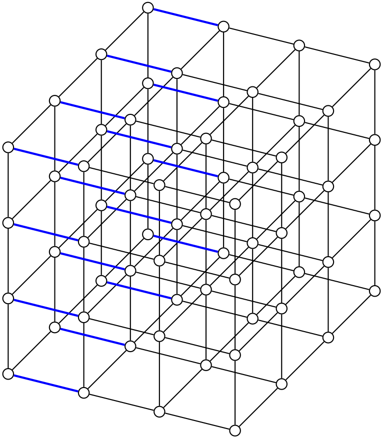
4-sequence for planar grids

3-dimensional grids



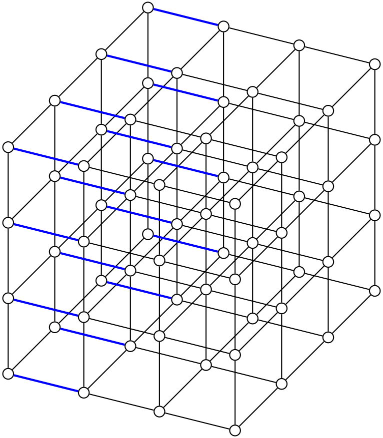
Contains arbitrary large clique minors

3-dimensional grids



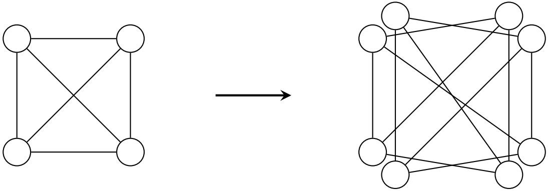
Contract the blue edges in any order $\rightarrow$ 12-sequence

3-dimensional grids



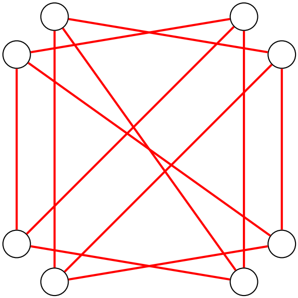
The d-dimensional grid has twin-width $\leqslant 4d$ (even $3d$)

2-lifts, expanders with bounded twin-width



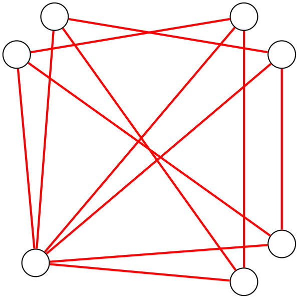
split each vertex in 2, replace each edge by 1 of the 2 matchings

2-lifts, expanders with bounded twin-width



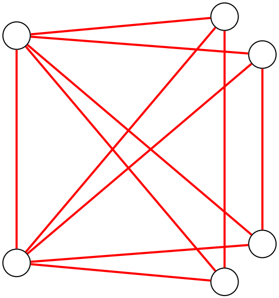
Iterated 2-lifts of K_4 have twin-width at most 6

2-lifts, expanders with bounded twin-width



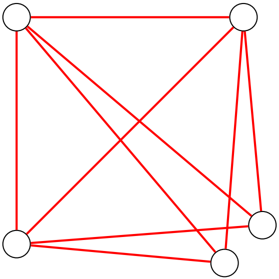
Iterated 2-lifts of K_4 have twin-width at most 6

2-lifts, expanders with bounded twin-width



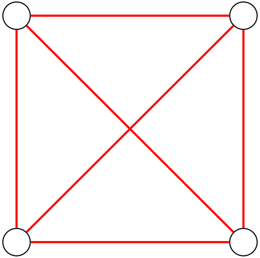
Iterated 2-lifts of K_4 have twin-width at most 6

2-lifts, expanders with bounded twin-width



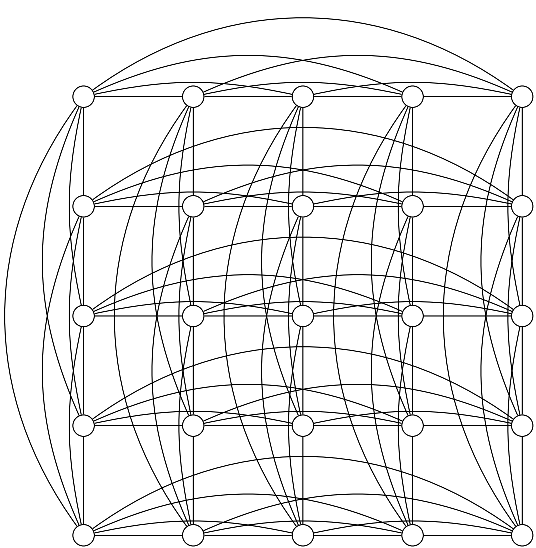
Iterated 2-lifts of K_4 have twin-width at most 6

2-lifts, expanders with bounded twin-width



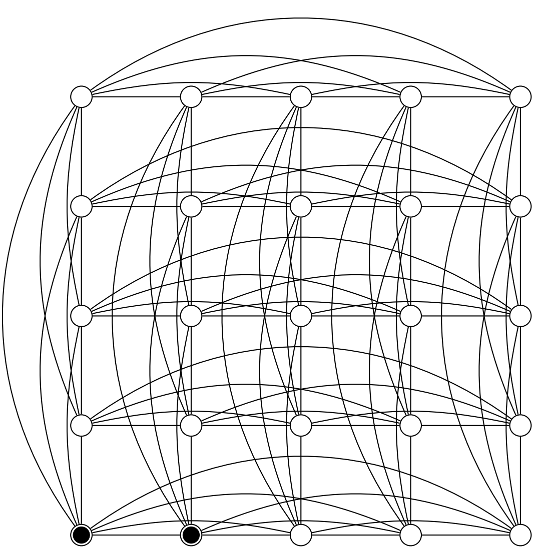
Iterated 2-lifts of K_4 have twin-width at most 6

First example of unbounded twin-width



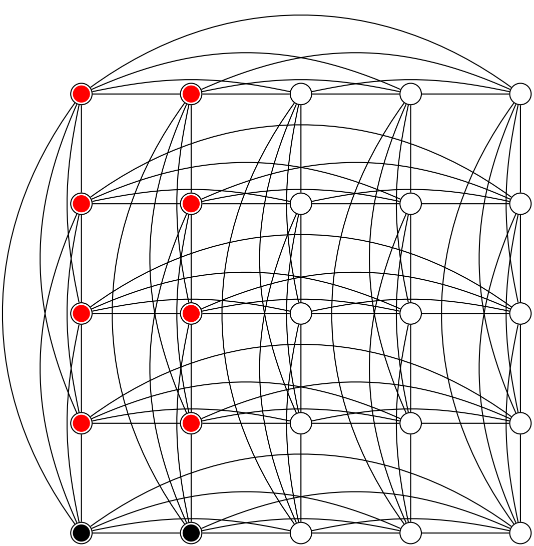
Line graph of a biclique a.k.a. rook graph

First example of unbounded twin-width



No pair of near twins

First example of unbounded twin-width



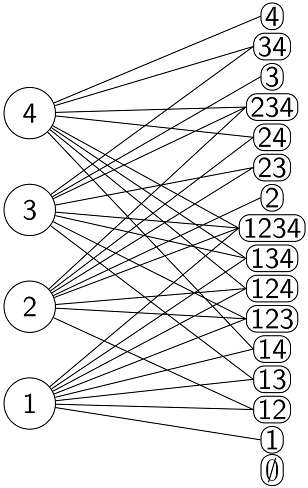
No pair of near twins

Universal bipartite graph

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

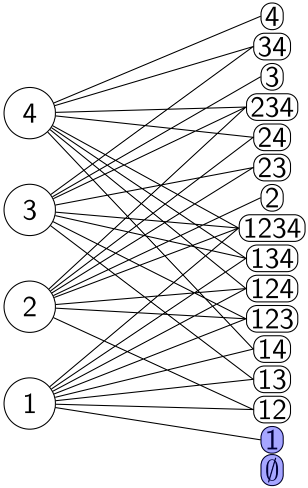
Universal bipartite graph

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



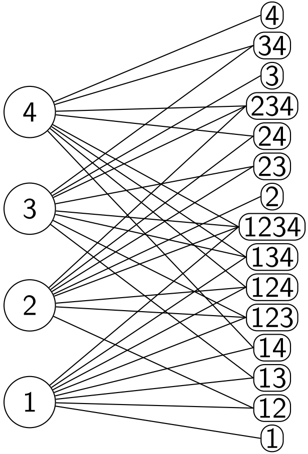
Universal bipartite graph

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



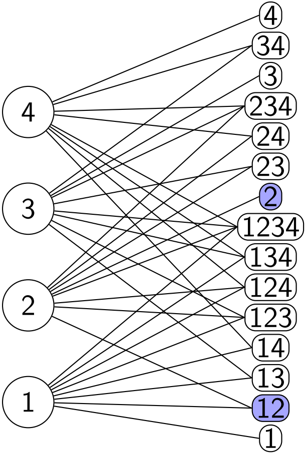
Universal bipartite graph

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



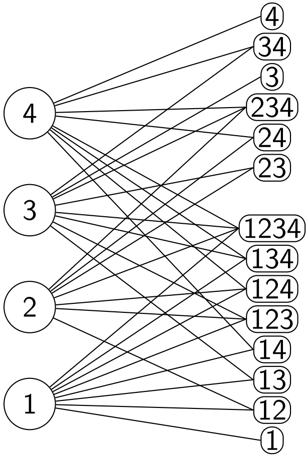
Universal bipartite graph

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



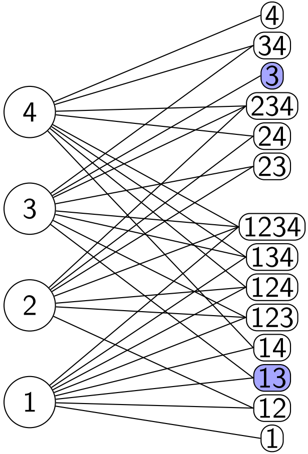
Universal bipartite graph

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



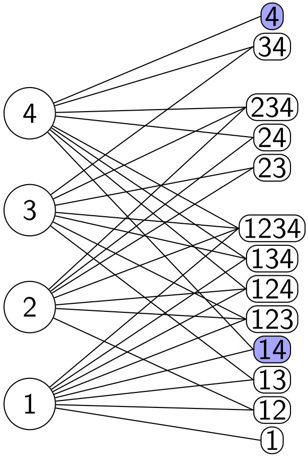
Universal bipartite graph

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



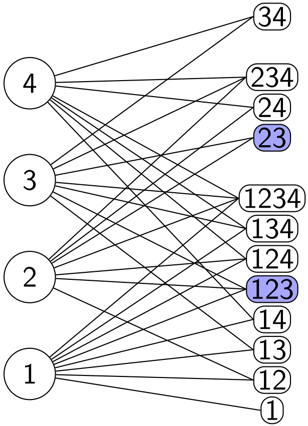
Universal bipartite graph

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



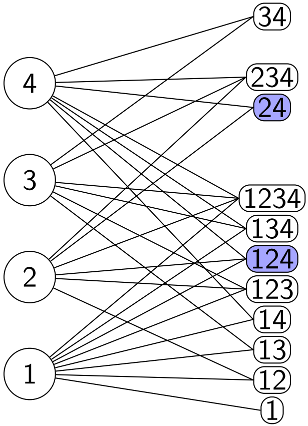
Universal bipartite graph

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



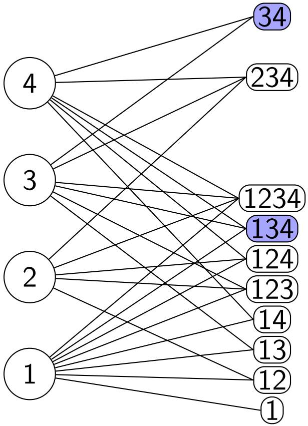
Universal bipartite graph

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



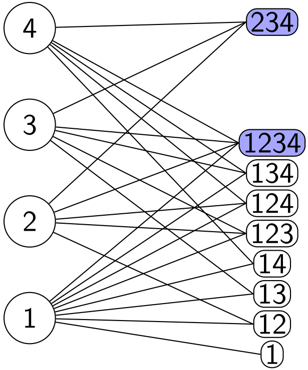
Universal bipartite graph

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



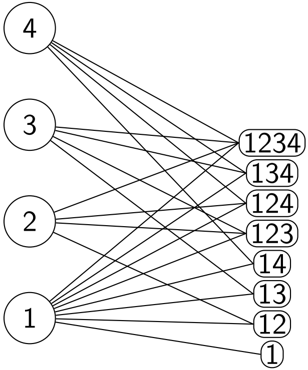
Universal bipartite graph

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



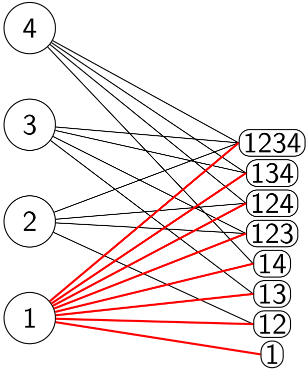
Universal bipartite graph

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



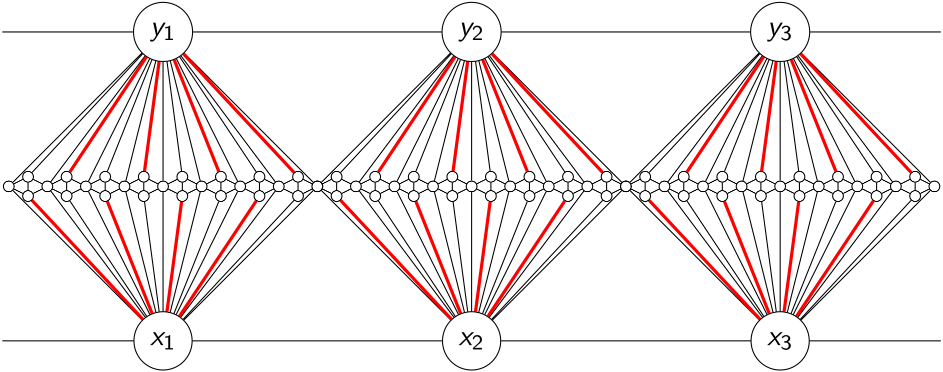
Universal bipartite graph

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



Planar graphs?

Planar graphs?



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

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

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

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 $\exists \sigma$ *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* $tww(G) = 2^{2^{O(t)}}$.

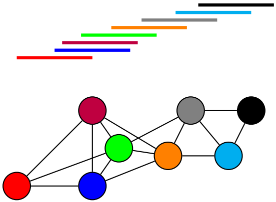
Now to bound the twin-width of a class $\mathcal{C}$:

1) Find a *good* vertex-ordering procedure

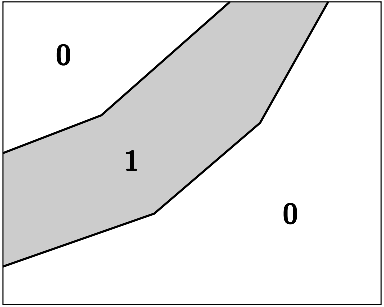
2) Argue that, in this order, a t-mixed minor would contradict the structure of $\mathcal{C}$

Unit interval graphs

Intersection graph of unit segments on the real line

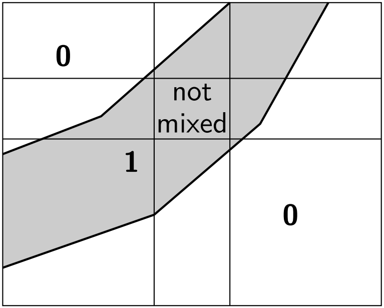


Unit interval graphs



order by left endpoints

Unit interval graphs



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

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

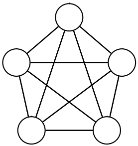
Graph minors

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

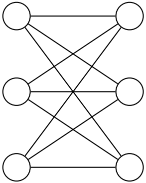
A graph G is *H-minor free* if H is not a minor of G

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

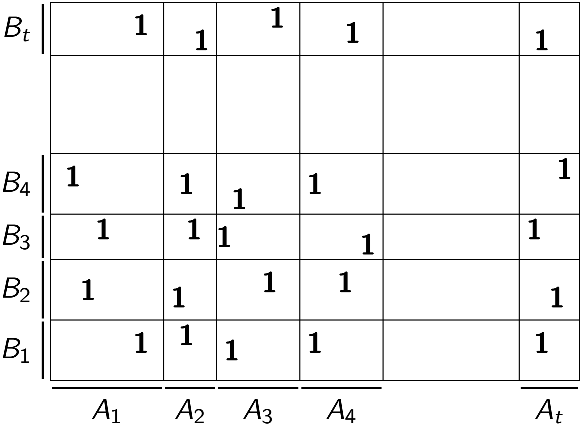


$K_{3,3}$

Bounded twin-width – K_t-minor free graphs

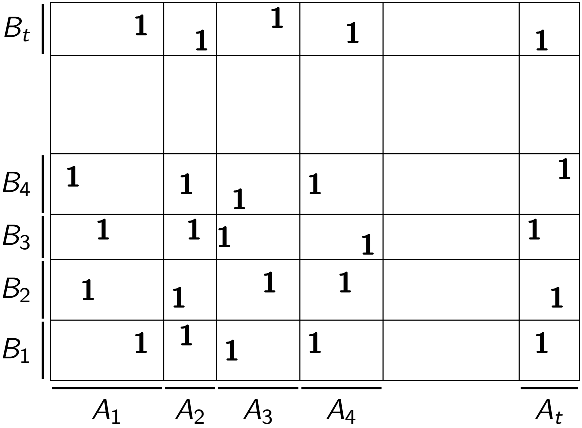
Given a hamiltonian path, we would just use this order

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 (B., Geniet, Kim, Thomassé, Watrigant '20 & '21)

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 (B., Geniet, Kim, Thomassé, Watrigant '20 & '21)

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?

k-Independent Set given a $d = O(1)$-sequence

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

Algorithm: **For every connected subset D of size at most k of the red graph of every G_i, store in $T[D, i]$ one largest independent set in $G\langle D \rangle$ intersecting every vertex of D.**

k-Independent Set given a $d = O(1)$-sequence

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

Algorithm: **For every connected subset D of size at most k of the red graph of every G_i, store in $T[D, i]$ one largest independent set in $G\langle D \rangle$ intersecting every vertex of D.**

Initialization: $T[\{v\}, n] = \{v\}$

End: $T[\{V(G)\}, 1] =$ IS of size at least k or largest IS in G

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

k-Independent Set given a $d = O(1)$-sequence

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

Algorithm: **For every connected subset D of size at most k of the red graph of every G_i, store in $T[D, i]$ one largest independent set in $G\langle D\rangle$ intersecting every vertex of D.**

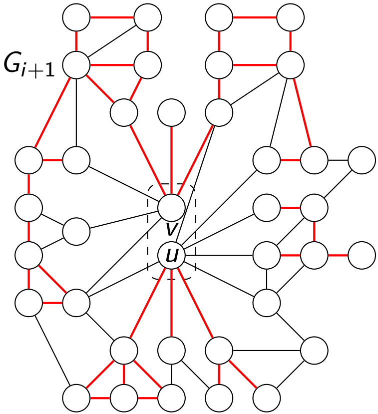
Initialization: $T[\{v\}, n] = \{v\}$

End: $T[\{V(G)\}, 1] =$ IS of size at least k or largest IS in G

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

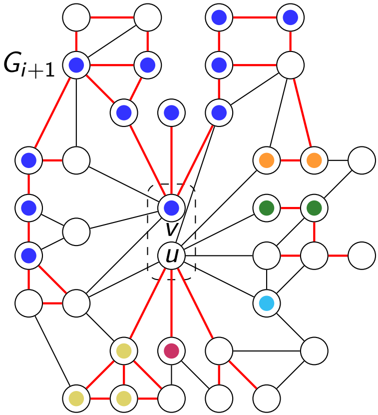
How to compute $T[D, i]$ from all the $T[D', i+1]$?

k-Independent Set: Update of partial solutions



Best partial solution inhabiting •?

k-Independent Set: Update of partial solutions



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

Formulas, sentences, and 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: $G \models \varphi$?

Formulas, sentences, and 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:

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

$G \models \varphi? \Leftrightarrow$

Formulas, sentences, and 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:

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

$G \models \varphi? \Leftrightarrow$ k-Dominating Set

Formulas, sentences, and 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:

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

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

Formulas, sentences, and model checking

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

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

Formulas, sentences, and 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:

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

$G \models \varphi? \Leftrightarrow$

Formulas, sentences, and 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

FO model checking on graphs of bounded twin-width

The previous algorithm generalizes to:

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.

χ-boundedness

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

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

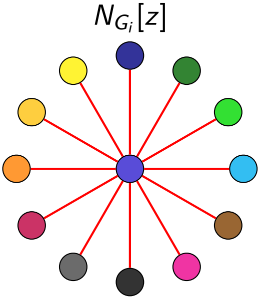
Every twin-width class is χ-bounded.
More precisely, every graph G of twin-width at most d admits a proper $(d+2)^{\omega(G)-1}$-coloring.

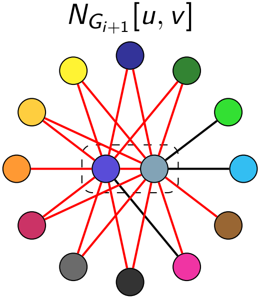
$d + 2$-coloring in the triangle-free case

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.**

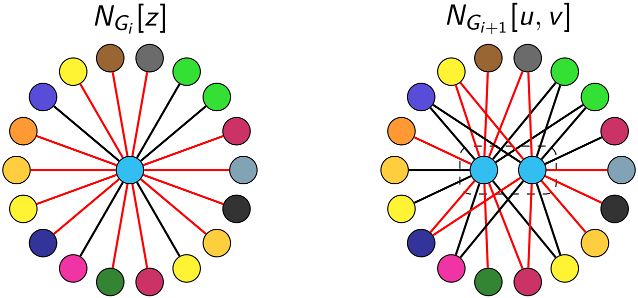




z has only red incident edges $\rightarrow$ $d + 2$-nd color available to v

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

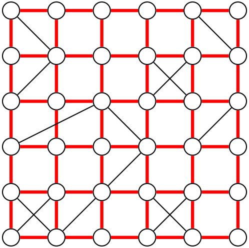
Algorithm: **Start from $G_1 = K_1$, color its unique vertex 1, and rewind the d-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.**



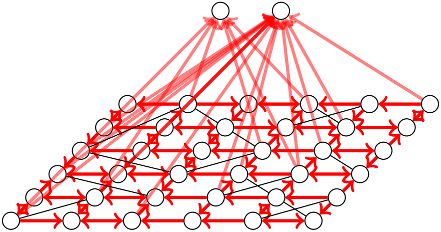
z incident to at least one black edge $\rightarrow$ non-edge between u and v

Perhaps contraction sequences are interesting independently of twin-width?

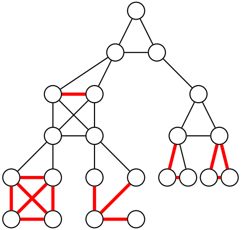
Different conditions imposed in the sequence of red graphs



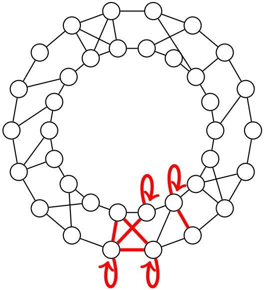
bd degree: defines bd twin-width



bd outdegree: defines bd oriented twin-width



bd component: redefines bd cliquewidth



bd #edges: redefines bd linear cliquewidth

Reduced parameters

A graph class has bounded reduced X if all its members admit a contraction sequence whose red graphs have bounded X

Reduced parameters

A graph class has bounded reduced X if all its members admit a contraction sequence whose red graphs have bounded X

red graphs have bounded ...	characterize bounded ...
degree	**twin-width**
component size	**cliquewidth**
number of edges*	linear cliquewidth
outdegree	(oriented) twin-width
degree + treewidth	?
cutwidth	?
bandwidth	?

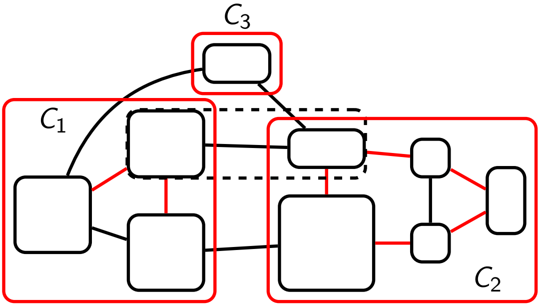
?'s = strict hierarchy of classes interpolating between bounded cliquewidth and bounded twin-width

Is it easier to design algorithms via this characterization?

Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d

Is it easier to design algorithms via this characterization?

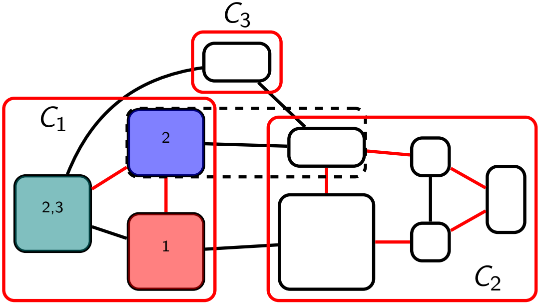
Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



For every red component C keep every profile $V(C) \to 2^{\{1,2,3\}} \setminus \{\emptyset\}$ realizable by a proper 3-coloring of $G\langle C \rangle$

Is it easier to design algorithms via this characterization?

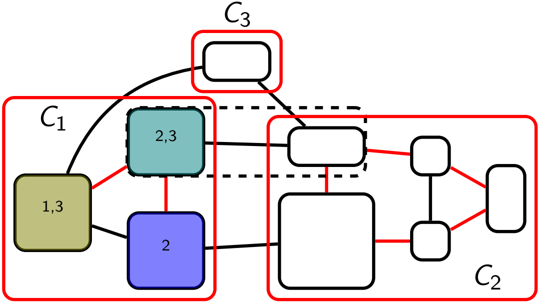
Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



For every red component C keep every profile $V(C) \to 2^{\{1,2,3\}} \setminus \{\emptyset\}$ realizable by a proper 3-coloring of $G\langle C \rangle$

Is it easier to design algorithms via this characterization?

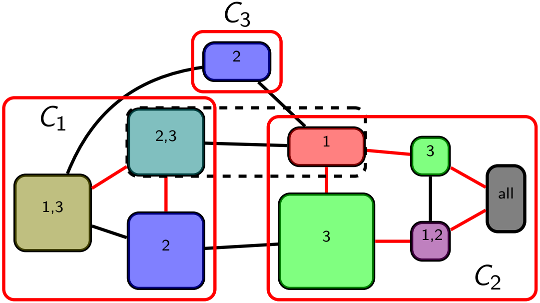
Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



For every red component C keep every profile $V(C) \to 2^{\{1,2,3\}} \setminus \{\emptyset\}$ realizable by a proper 3-coloring of $G\langle C \rangle$

Is it easier to design algorithms via this characterization?

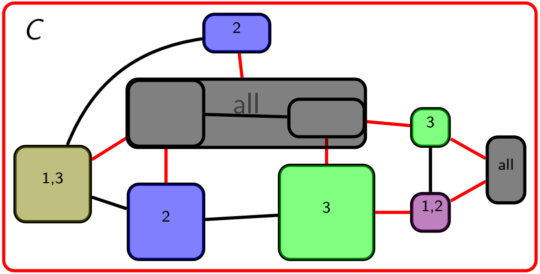
Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



Some tuples of the at most $d+1$ profiles corresponding to merging red components are compatible

Is it easier to design algorithms via this characterization?

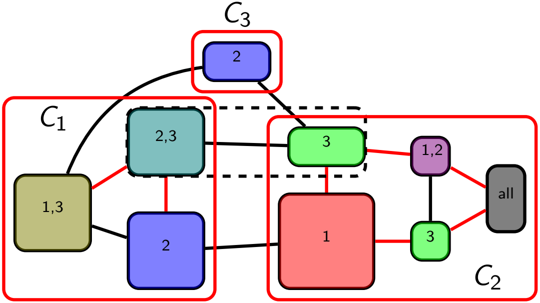
Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



Some tuples of the at most $d+1$ profiles corresponding to merging red components are compatible

Is it easier to design algorithms via this characterization?

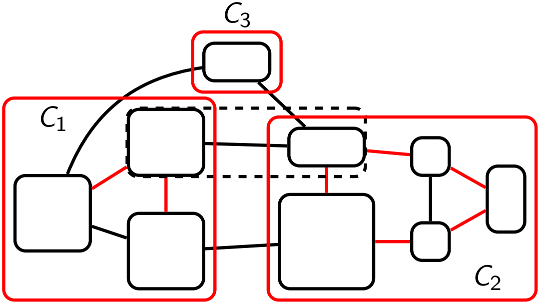
Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



Some tuples of the at most $d + 1$ profiles corresponding to merging red components are incompatible

Is it easier to design algorithms via this characterization?

Solve 3-Coloring on a graph G with a contraction sequence s.t. all red graphs have components of size at most d



Initialization: time $3n$
Update: time $7^d d^2$
Total: time $7^d d^2 n$
End: still a profile on the single vertex *containing* the whole graph?

Courcelle's theorems

We can recast and prove:

Theorem (Courcelle, Makowsky, Rotics '00)

MSO model checking can be solved in time $f(|\varphi|, d) \cdot |V(G)|$ given a witness that the clique-width/component twin-width of the input G is at most d.

which *generalizes*

Theorem (Courcelle '90)

MSO model checking can be solved in time $f(|\varphi|, t) \cdot |V(G)|$ on graphs G of treewidth at most t.

Concluding remarks

Contraction sequences give:

- twin-width for which first-order logic is tractable
- a new and unifying perspective on older width parameters

Concluding remarks

Contraction sequences give:

- twin-width for which first-order logic is tractable
- a new and unifying perspective on older width parameters

Main open question:
an efficient algorithm to approximate twin-width

Concluding remarks

Contraction sequences give:

- twin-width for which first-order logic is tractable
- a new and unifying perspective on older width parameters

Main open question:
an efficient algorithm to approximate twin-width

Thank you for your attention!