Algorithmic Meta-Theorems

192.122

WS21/22

Jan Dreier

dreier@ac.tuwien.ac.at

ALGORITHMS AND COMPLEXITY GROUP

Informatics

Announcements

- Today last lecture, Friday was last exercise.
- 30 min overtime today.
- Feedback on Tuwel is highly appreciated (short anonymous survey).
- Oral exam: in January/February
- There will be a question session before the oral exam.
- Write email until end of year to dreier@ac.tuwien.ac.at if you want to participate in the oral exam.

Main Result for Sparse Graphs

Today, we finish the proof of the following theorem.

Theorem (Dvořák, Král, Thomas 2013)

Let $\mathcal{C}$ be a graph class with bounded expansion. There exists a function f such that for every FO formula φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

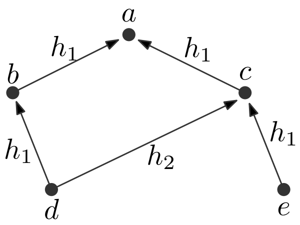
And then briefly talk about dense graphs.

Functional Graphs

A *functional graph* $\vec{G}$ is a structure with signature $\tau = \{h_1, h_2, \ldots, R_1, R_2, \ldots, Q_1, Q_2, \ldots\}$ where

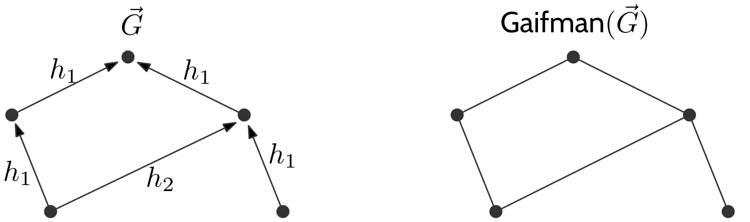
- $h_i \colon V \to V$ are unary functions
- $R_i \subseteq V$ are unary relations
- $Q_i \in \{0, 1\}$ are nullary relations

$$h_1(d) = b$$
$$h_2(d) = c$$



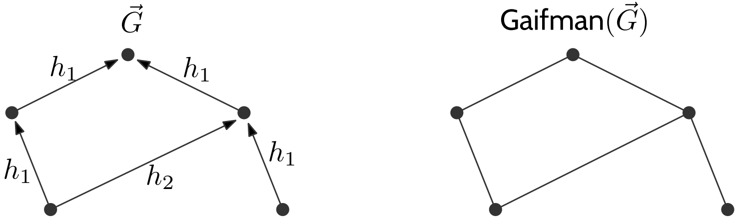
Classes

For a functional graph $\vec{G}$, the graph Gaifman$(\vec{G})$ has the same vertex set and edges uv iff $h_i(u) = v$ or $h_i(v) = u$ for some i.



Classes

For a functional graph $\vec{G}$, the graph Gaifman$(\vec{G})$ has the same vertex set and edges uv iff $h_i(u) = v$ or $h_i(v) = u$ for some i.



We say a class of functional graphs has *bounded expansion* if the class of their Gaifman graphs has.

Quantifier Elimination

$$\begin{array}{ccc} \vec{G} & & \vec{G}' \\ & \longrightarrow & \\ \exists y\, \varphi(y\bar{x}) & & \varphi'(\bar{x}) \end{array}$$

$$\begin{array}{ccc} \vec{G} & & \vec{G}' \\ \exists y\, \varphi(y\bar{x}) & \rightarrow & \varphi'(\bar{x}) \end{array}$$

Should still be short. Length depends only on φ.

Quantifier Elimination

Should still be sparse. Same Gaifman graph as $\vec{G}$.

$$\begin{matrix} \vec{G} & & \vec{G}' \\ \exists y\, \varphi(y\bar{x}) & \longrightarrow & \varphi'(\bar{x}) \end{matrix}$$

Should still be short. Length depends only on φ.

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

For every formula $\exists y\, \varphi(y\bar{x})$ where φ is quantifier free there exists a quantifier-free formula $\varphi'(\bar{x})$ as follows:

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

For every formula $\exists y\, \varphi(y\bar{x})$ where φ is quantifier free there exists a quantifier-free formula $\varphi'(\bar{x})$ as follows:

For every $\vec{G} \in \vec{\mathcal{C}}$ one can compute in time $O(|\vec{G}|)$ a functional graph $\vec{G}'$ with the same Gaifman graph as $\vec{G}$ and

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

For every formula $\exists y\, \varphi(y\bar{x})$ where φ is quantifier free there exists a quantifier-free formula $\varphi'(\bar{x})$ as follows:

For every $\vec{G} \in \vec{\mathcal{C}}$ one can compute in time $O(|\vec{G}|)$ a functional graph $\vec{G}'$ with the same Gaifman graph as $\vec{G}$ and

$$\vec{G} \models \exists y\, \varphi(y\bar{v}) \quad \text{iff} \quad \vec{G}' \models \varphi'(\bar{v}) \qquad \text{for all } \bar{v} \in V(\vec{G})^{|\bar{x}|}.$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G} \models \exists x_1 \left(\qquad\qquad\qquad\qquad\qquad\qquad \right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\qquad\qquad\qquad\qquad \right)\right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\exists x_3 \qquad\qquad\qquad\qquad \right)\right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\exists x_3 \overbrace{\varphi(x_1 x_2 x_3)}^{\text{quantifer-free}} \right) \right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\underbrace{\exists x_3 \; \overbrace{\varphi(x_1 x_2 x_3)}^{\text{quantifer-free}}}_{\text{replace with quantifier-free}} \right)\right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G}' \models \exists x_1 \left(\forall x_2 \; \overbrace{\varphi'(x_1 x_2)}^{\text{quantifier-free}} \right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G}' \models \exists x_1 \left(\underbrace{\forall x_2 \quad \overbrace{\varphi'(x_1 x_2)}^{\text{quantifier-free}}}_{\text{replace with quantifier-free}} \right)$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G}'' \models \exists x_1 \quad \overbrace{\varphi''(x_1)}^{\text{quantifier-free}}$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G}'' \models \underbrace{\exists x_1 \quad \overbrace{\varphi''(x_1)}^{\text{quantifier-free}}}_{\text{replace with quantifier-free}}$$

Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G}''' \models \overbrace{\varphi'''}^{\text{quantifier-free}}$$

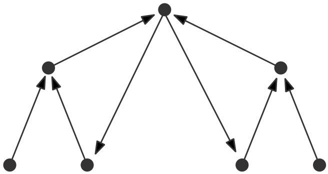
Roadmap

Roadmap: Construct quantifier elimination for graph classes of increasing complexity.

- forests of bounded depth ✓
- bounded treedepth
- bounded expansion

Bounded Treedepth

Let $\mathcal{C}$ be a class whose Gaifman graphs have *bounded treedepth*. Then $\mathcal{C}$ has a quantifier elimination procedure.

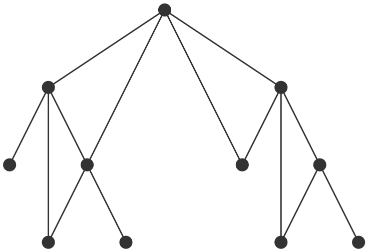


Bounded Treedepth

bounded
treedepth

$$\vec{G}$$

$$\exists y \, \varphi(y\bar{x})$$

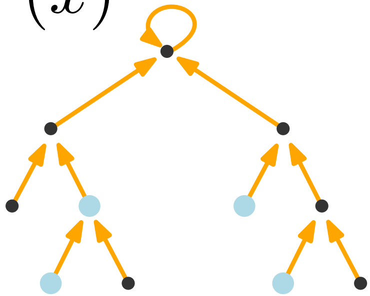


Bounded Treedepth

bounded treedepth → $\vec{G}$

forest of bounded depth → $\vec{G}'$

$$\exists y\, \varphi(y\bar{x}) \quad \longrightarrow \quad \exists y\, \varphi'(\bar{x})$$



Bounded Treedepth

bounded treedepth

$$\vec{G}$$

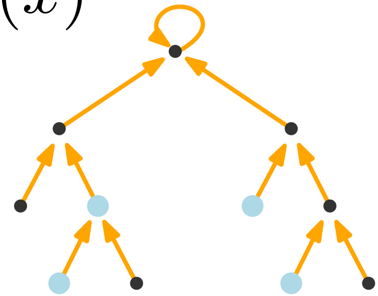
$$\longrightarrow$$

forest of bounded depth

$$\vec{G}''$$

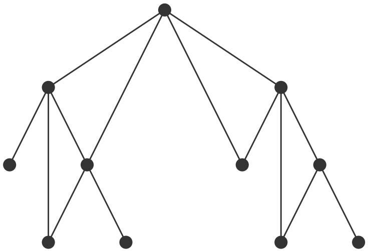
$$\exists y\, \varphi(y\bar{x}) \qquad \varphi''(\bar{x})$$

Use previous theorem for quantifier elimination on forests.



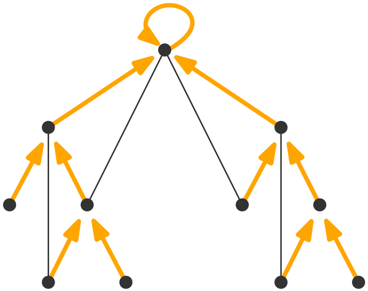
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$.



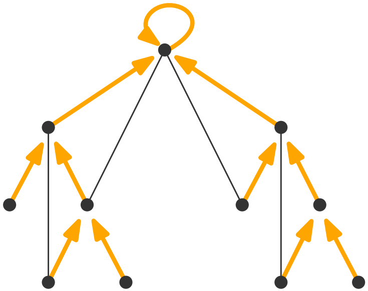
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional.



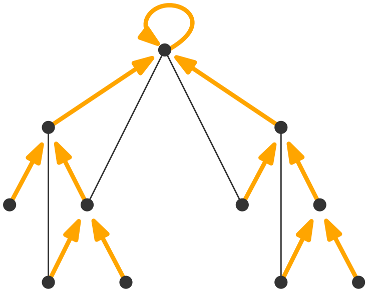
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d.



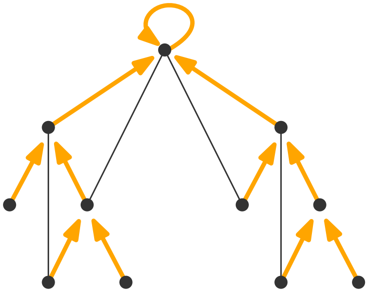
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.



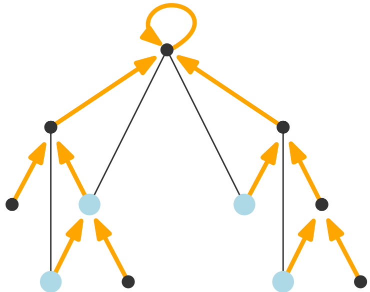
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.
- All edges go between ancestors in the tree.



Bounded Treedepth

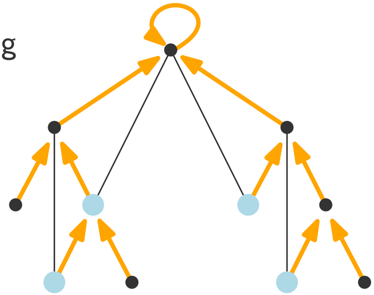
- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.
- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.



Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.
- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.

$P_{i,j,\updownarrow}(v)$: "There is edge from v to parent$^i(v)$ labeled with j going upwards/downwards".

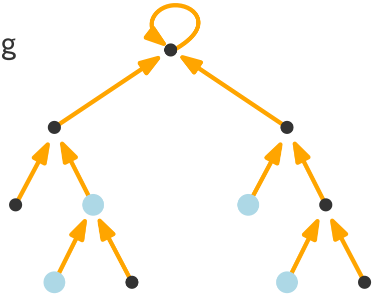


Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.
- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.

  $P_{i,j,\updownarrow}(v)$: "There is edge from v to parent$^i(v)$ labeled with j going upwards/downwards".

- This tree fully encodes $\vec{G}$.

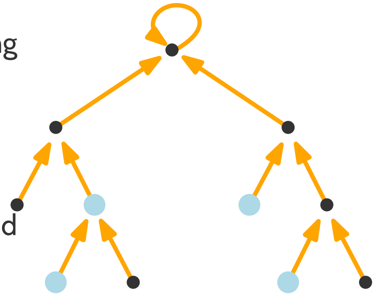


Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.
- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.

  $P_{i,j,\updownarrow}(v)$: “There is edge from v to parent$^i(v)$ labeled with j going upwards/downwards”.

- This tree fully encodes $\vec{G}$.
- Replace atoms $f_j(x) = y$ by guessing tree-relationship and checking the new predicates.

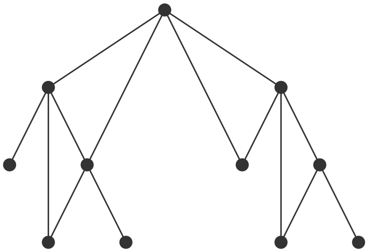


Bounded Treedepth

bounded
treedepth

$$\vec{G}$$

$$\exists y\, \varphi(y\bar{x})$$



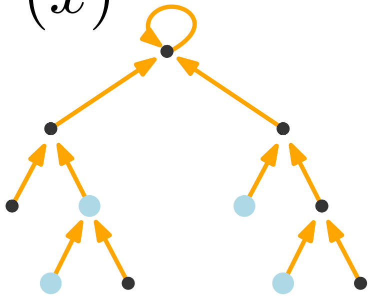
Bounded Treedepth

bounded treedepth

$$\vec{G} \longrightarrow \vec{G}'$$

forest of bounded depth

$$\exists y\, \varphi(y\bar{x}) \qquad \exists y\, \varphi'(\bar{x})$$



Bounded Treedepth

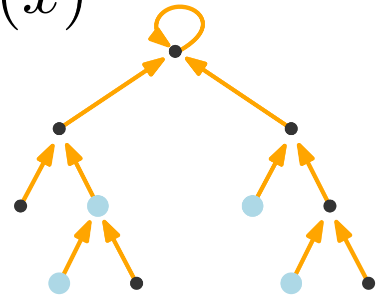
bounded treedepth

forest of bounded depth

$$\vec{G} \longrightarrow \vec{G}''$$

$$\exists y\, \varphi(y\bar{x}) \qquad \varphi''(\bar{x})$$

Use previous theorem for quantifier elimination on forests.



The last step.

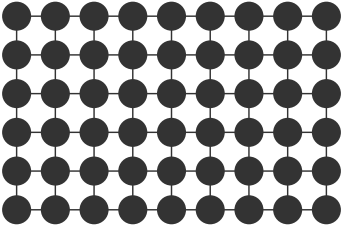
Let $\mathcal{C}$ be a class with bounded expansion. Then $\mathcal{C}$ has a quantifier elimination procedure.

Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.

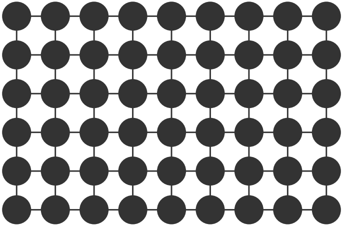
Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



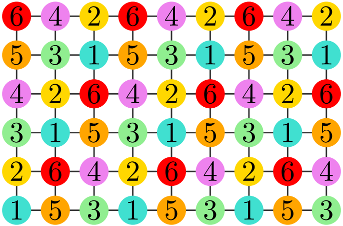
Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



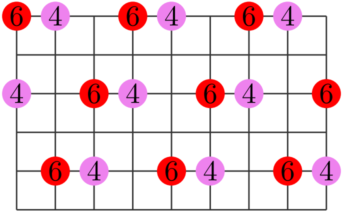
Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



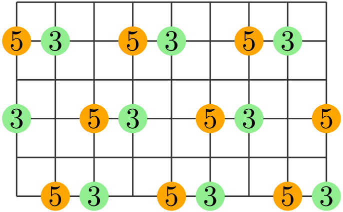
Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



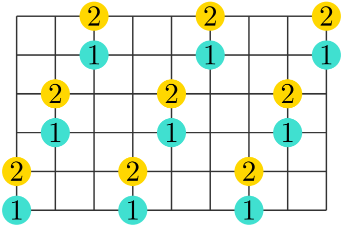
Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



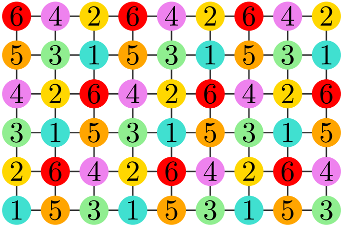
Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



Low Treedepth Colorings

One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of p colors induces a graph with treedepth $\leq p$.



Bounded Expansion

We are given a formula $\exists y \varphi(\bar{x})$ where φ is quantifier-free.

Bounded Expansion

We are given a formula $\exists y \varphi(\bar{x})$ where φ is quantifier-free.

Replace all function applications such as $p(g(x_i)) = x_j$ with directed labeled edges such as $\exists a\, x_i \underset{g}{\rightsquigarrow} a \wedge a \underset{p}{\rightsquigarrow} x_j$.

Bounded Expansion

We are given a formula $\exists y \varphi(\bar{x})$ where φ is quantifier-free.

Replace all function applications such as $p(g(x_i)) = x_j$ with directed labeled edges such as $\exists a \, x_i \underset{g}{\rightsquigarrow} a \wedge a \underset{p}{\rightsquigarrow} x_j$.

The result is a formula $\exists \bar{y} \, \psi(\bar{y}\bar{x})$ on a normal (non-functional) directed edge-labeled graph G.

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

Let Λ be the colors of a low-treedepth coloring of G where every subgraph on $p = |\bar{y}\bar{x}|$ colors has treedepth $\leq p$.

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge$$

Let Λ be the colors of a low-treedepth coloring of G where every subgraph on $p = |\bar{y}\bar{x}|$ colors has treedepth $\leq p$.

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S| = p}} \text{colors}(\bar{v}) \subseteq S \wedge G[S] \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

Let Λ be the colors of a low-treedepth coloring of G where every subgraph on $p = |\bar{y}\bar{x}|$ colors has treedepth $\leq p$.

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge G[S] \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

For all graphs $G[S]$ (of treedepth $\leq p$) we perform quantifier elimination.

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge G[S] \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge \vec{G}'_S \models \psi_S(\bar{v}) \quad \text{iff}$$

For all graphs $G[S]$ (of treedepth $\leq p$) we perform quantifier elimination.

Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge G[S] \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge \vec{G}' \models \psi_S(\bar{v}) \quad \text{iff}$$

For all graphs $G[S]$ (of treedepth $\leq p$) we perform quantifier elimination. And stack the graphs on top of each other.

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge G[S] \models \exists \bar{y}\, \psi(\bar{y}\bar{v}) \quad \text{iff}$$

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge \vec{G}' \models \psi_S(\bar{v}) \quad \text{iff}$$

$$\vec{G}' \models \bigvee_{\substack{S \subseteq \Lambda \\ |S|=p}} \text{colors}(\bar{v}) \subseteq S \wedge \psi_S(\bar{v}).$$

Finally, pull out the graph.

Summary

This completes the proof. We have solved the model-checking problem on bounded expansion by performing quantifier elimination on trees and lifting it up using low treedepth colorings.

A Note on Nowhere Dense Classes

General rule: Things that work for bounded expansion also work for nowhere dense, but in an uglier way.

Comparison

Bounded Expansion

Nowhere Dense

Comparison

Bounded Expansion

Nowhere Dense

p-treedepth colorings with $f(p)$ colors ✓

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^{\varepsilon}$ colors for all $\varepsilon > 0$ ✓

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^{\varepsilon}$ colors for all $\varepsilon > 0$ ✓

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon > 0$ ✓

Enumerate p-subsets in time $\binom{f(\varepsilon,p)n^\varepsilon}{p}$

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon > 0$ ✓

Enumerate p-subsets in time $\binom{f(\varepsilon,p)n^\varepsilon}{p}$
$\leq f(\varepsilon, p)n^{\varepsilon p}$

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^{\varepsilon}$ colors for all $\varepsilon > 0$ ✓

Enumerate p-subsets in time $\binom{f(\varepsilon,p)n^{\varepsilon}}{p}$
$\leq f(\varepsilon, p)n^{\varepsilon p} = f(\varepsilon'/p, p)n^{\varepsilon'}$ ✓

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Do something very expensive in time $2^{2^{f(p)}}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^{\varepsilon}$ colors for all $\varepsilon > 0$ ✓

Enumerate p-subsets in time $\binom{f(\varepsilon,p)n^{\varepsilon}}{p}$
$\leq f(\varepsilon, p)n^{\varepsilon p} = f(\varepsilon'/p, p)n^{\varepsilon'}$ ✓

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Do something very expensive in time $2^{2^{f(p)}}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^{\varepsilon}$ colors for all $\varepsilon > 0$ ✓

Enumerate p-subsets in time $\binom{f(\varepsilon,p)n^{\varepsilon}}{p}$
$\leq f(\varepsilon, p)n^{\varepsilon p} = f(\varepsilon'/p, p)n^{\varepsilon'}$ ✓

Do something very expensive in time $2^{2^{f(\varepsilon/p,p)n^{\varepsilon}}}$

Comparison

Bounded Expansion

p-treedepth colorings with $f(p)$ colors ✓

Enumerate p-subsets in time $\binom{f(p)}{p}$ ✓

Do something very expensive in time $2^{2^{f(p)}}$ ✓

Nowhere Dense

p-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon > 0$ ✓

Enumerate p-subsets in time $\binom{f(\varepsilon,p)n^\varepsilon}{p}$
$\leq f(\varepsilon, p)n^{\varepsilon p} = f(\varepsilon'/p, p)n^{\varepsilon'}$ ✓

Do something very expensive in time $2^{2^{f(\varepsilon/p,p)n^\varepsilon}}$
$\geq 2^{2^{\log(n)}} = 2^n$ ✗

Beyond Nowhere Dense?

We have gradually defined more and more general sparse graph classes. The most general so far are nowhere dense classes.

Beyond Nowhere Dense?

We have gradually defined more and more general sparse graph classes. The most general so far are nowhere dense classes.

Are there any more interesting sparse graph classes beyond nowhere dense?

Beyond Nowhere Dense?

We have gradually defined more and more general sparse graph classes. The most general so far are nowhere dense classes.

Are there any more interesting sparse graph classes beyond nowhere dense?

We show the answer is no! We have found the most general sparse graph classes on which first-order problems are tractable!

Precise Chacaterization

A graph class is *monotone* if it is closed under removing vertices or edges. This is a natural assumption for sparse graphs.

Precise Chacaterization

A graph class is *monotone* if it is closed under removing vertices or edges. This is a natural assumption for sparse graphs.

Theorem (Grohe, Kreuzer, Siebertz 2017)

For every monotone graph class $\mathcal{C}$ holds $\mathcal{C}$ is nowhere dense iff the first-order model-checking problem on $\mathcal{C}$ is fpt (assuming FPT $\neq$ AW[$*$]).

Precise Chacaterization

A graph class is *monotone* if it is closed under removing vertices or edges. This is a natural assumption for sparse graphs.

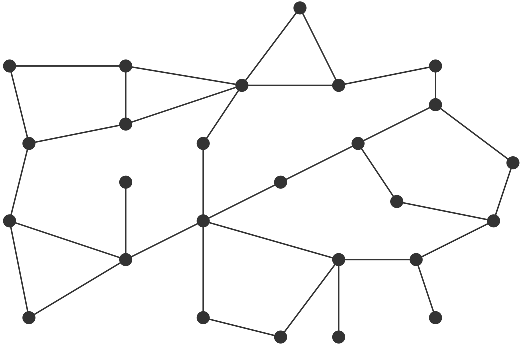
Theorem (Grohe, Kreuzer, Siebertz 2017)

For every monotone graph class $\mathcal{C}$ holds $\mathcal{C}$ is nowhere dense iff the first-order model-checking problem on $\mathcal{C}$ is fpt (assuming FPT $\neq$ AW[$*$]).

We discussed the forward implication already. We will prove the backward implication.

Shallow Topological Minors

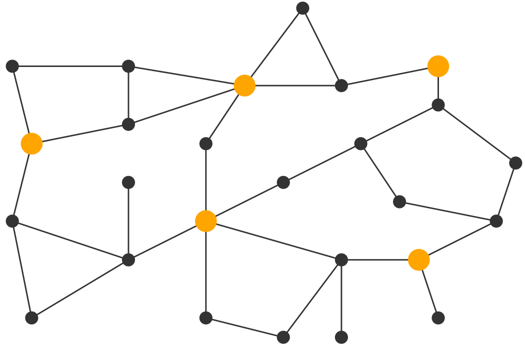
H is an *depth-r topological minor* of G ($H \preccurlyeq_r^{\mathrm{top}} G$) if H can be built from G by



Shallow Topological Minors

H is an *depth-r topological minor* of G ($H \preccurlyeq_r^{\mathrm{top}} G$) if H can be built from G by

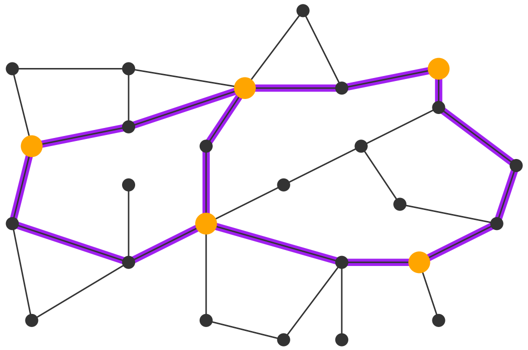
- picking some *nails*,



Shallow Topological Minors

H is an *depth-r topological minor* of G ($H \preccurlyeq_r^{\mathrm{top}} G$) if H can be built from G by

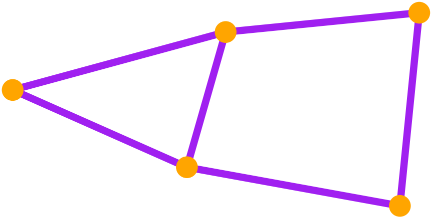
- picking some *nails*,
- picking internally vertex disjoint paths of length at most $2r + 1$ between nails,



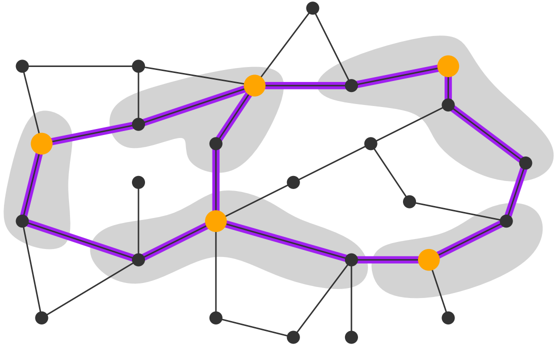
Shallow Topological Minors

H is an *depth-r topological minor* of G ($H \preccurlyeq_r^{\text{top}} G$) if H can be built from G by

- picking some *nails*,
- picking internally vertex disjoint paths of length at most $2r + 1$ between nails,
- contracting the paths.



If H is an depth-r topological minor then it also is a depth-r minor.



Topological Sparsity Measures

We can also measure sparsity using topological minors.

Topological Sparsity Measures

We can also measure sparsity using topological minors.

- $\nabla_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r \quad G\}$
- $\omega_r(G) = \max\{t \mid K_t \preccurlyeq_r \quad G\}$

Topological Sparsity Measures

We can also measure sparsity using topological minors.

- $\tilde{\nabla}_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r^{\text{top}} G\}$
- $\tilde{\omega}_r(G) = \max\{t \mid K_t \preccurlyeq_r^{\text{top}} G\}$

Topological Sparsity Measures

We can also measure sparsity using topological minors.

- $\tilde{\nabla}_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r^{\text{top}} G\}$
- $\tilde{\omega}_r(G) = \max\{t \mid K_t \preccurlyeq_r^{\text{top}} G\}$

As we saw on the last slide

- $\tilde{\nabla}_r(G) \leq \nabla_r(G)$
- $\tilde{\omega}_r(G) \leq \omega_r(G)$

Topological Sparsity Measures

We can also measure sparsity using topological minors.

- $\tilde{\nabla}_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r^{\text{top}} G\}$
- $\tilde{\omega}_r(G) = \max\{t \mid K_t \preccurlyeq_r^{\text{top}} G\}$

As we saw on the last slide

- $\tilde{\nabla}_r(G) \leq \nabla_r(G)$
- $\tilde{\omega}_r(G) \leq \omega_r(G)$

Surprisingly, also

- $\nabla_r(G) \leq 2^{d^2+3d+3}(\lceil\tilde{\nabla}_r(G)\rceil)^{(d+2)^2}$
- $\omega_r(G) \leq 1 + (\tilde{\omega}_{3r+1}(G))^{2d+2}$

Equivalent Definitions

Therefore it does not matter if we consider normal or topological shallow minors.

Bounded Expansion

A graph class $\mathcal{C}$ has bounded expansion if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\nabla_r(G) \leq f(r)$.

Nowhere Dense

A graph class $\mathcal{C}$ is nowhere dense if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\omega_r(G) \leq f(r)$.

Equivalent Definitions

Therefore it does not matter if we consider normal or topological shallow minors.

Bounded Expansion

A graph class $\mathcal{C}$ has bounded expansion if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\tilde{\nabla}_r(G) \leq f(r)$.

Nowhere Dense

A graph class $\mathcal{C}$ is nowhere dense if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\tilde{\omega}_r(G) \leq f(r)$.

Somewhere Dense

We can slightly restate this.

Nowhere Dense

A graph class $\mathcal{C}$ is nowhere dense if for all $r \in \mathbb{N}$ there exists $t \in \mathbb{N}$ such that for all $G \in \mathcal{C}$ we have $K_t \not\preccurlyeq_r^{\text{top}} G$.

Somewhere Dense

We can slightly restate this.

Nowhere Dense

A graph class $\mathcal{C}$ is nowhere dense if for all $r \in \mathbb{N}$ there exists $t \in \mathbb{N}$ such that for all $G \in \mathcal{C}$ we have $K_t \not\preccurlyeq_r^{\mathrm{top}} G$.

We say a graph class is *somewhere dense* if it is not nowhere dense:

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\mathrm{top}} G$.

Subdivisions of Arbitrary Subgraphs

We show the following.

Lemma

If a graph class is monotone and somewhere dense then first-order model-checking on this class is AW[$*$]-hard.

Subdivisions of Arbitrary Subgraphs

We need Ramsey's Theorem.

Lemma

For every $t, r \in \mathbb{N}$ there is $k \in \mathbb{N}$ such that an edge-coloring of K_k with r colors contains a monochromatic K_t.

Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\mathrm{top}} G$.

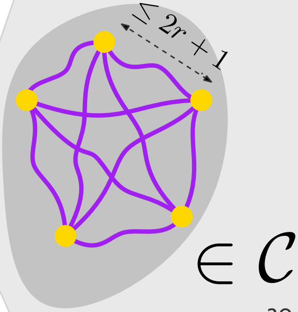
- Let $\mathcal{C}$ be monotone, somewhere dense.

Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\mathrm{top}} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick r s.t. for every t there is a graph in $\mathcal{C}$ with K_t as depth-r topological minor.

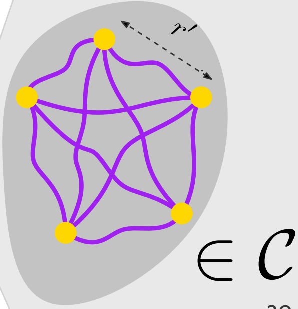


Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\text{top}} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick r s.t. for every t there is a graph in $\mathcal{C}$ with K_t as depth-r topological minor.
- Using Ramsey, we can enforce that every path has length *exactly* $r' \leq 2r + 1$.

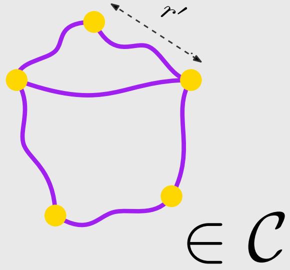


Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\text{top}} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick r s.t. for every t there is a graph in $\mathcal{C}$ with K_t as depth-r topological minor.
- Using Ramsey, we can enforce that every path has length *exactly* $r' \leq 2r + 1$.
- Since $\mathcal{C}$ is monotone, we can remove arbitrary edges and vertices.

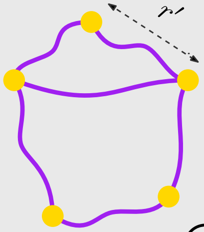


Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\text{top}} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick r s.t. for every t there is a graph in $\mathcal{C}$ with K_t as depth-r topological minor.
- Using Ramsey, we can enforce that every path has length *exactly* $r' \leq 2r + 1$.
- Since $\mathcal{C}$ is monotone, we can remove arbitrary edges and vertices.
- $\mathcal{C}$ contains r'-subdivisions of arbitrary graphs.



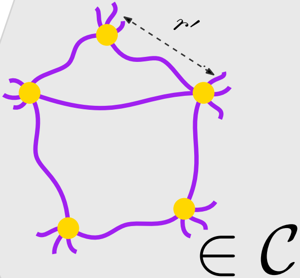
$\in \mathcal{C}$

Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preccurlyeq_r^{\text{top}} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick r s.t. for every t there is a graph in $\mathcal{C}$ with K_t as depth-r topological minor.
- Using Ramsey, we can enforce that every path has length *exactly* $r' \leq 2r + 1$.
- Since $\mathcal{C}$ is monotone, we can remove arbitrary edges and vertices.
- $\mathcal{C}$ contains r'-subdivisions of arbitrary graphs.
- We may even assume that the nails have some "hair" to distinguish them.



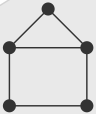
Reduction

We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[$*$]-hard.

Reduction

We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[$*$]-hard.

- Assume we want to know whether $G \models \varphi$.

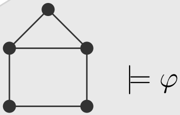


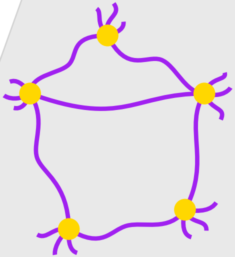
$\models \varphi$

Reduction

We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[$*$]-hard.

- Assume we want to know whether $G \models \varphi$.
- Subdivide each edge r' times and add hair. The result is in $\mathcal{C}$.

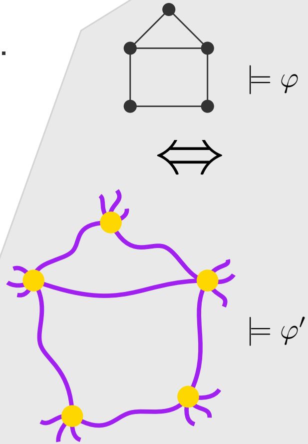




Reduction

We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[$*$]-hard.

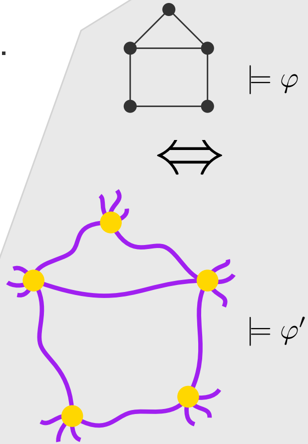
- Assume we want to know whether $G \models \varphi$.
- Subdivide each edge r' times and add hair. The result is in $\mathcal{C}$.
- Replace
  - $x \sim y$ with $\mathrm{dist}(x, y) \leq r'$.
  - $\exists x \xi$ with $\exists x\ \mathrm{degree}(x) \geq 3 \wedge \xi$
  - $\forall x \xi$ with $\forall x\ \mathrm{degree}(x) \geq 3 \rightarrow \xi$



Reduction

We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[$*$]-hard.

- Assume we want to know whether $G \models \varphi$.
- Subdivide each edge r' times and add hair. The result is in $\mathcal{C}$.
- Replace
  - $x \sim y$ with $\mathrm{dist}(x, y) \leq r'$.
  - $\exists x \xi$ with $\exists x$ $\mathrm{degree}(x) \geq 3 \wedge \xi$
  - $\forall x \xi$ with $\forall x$ $\mathrm{degree}(x) \geq 3 \rightarrow \xi$
- Since the modified graph is in $\mathcal{C}$, the modified formula can be evaluated in fpt time.



Summary

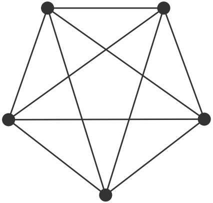
For monotone graph classes, we cannot go beyond nowhere dense classes. Nowhere dense classes are a natural definition of “sparse graphs”.

Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques.

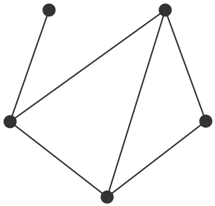
Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.



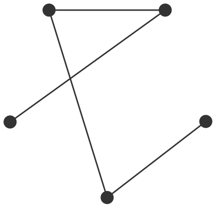
Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.



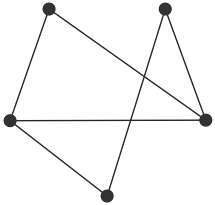
Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.



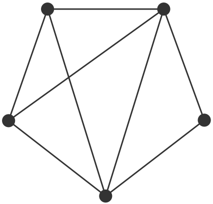
Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.



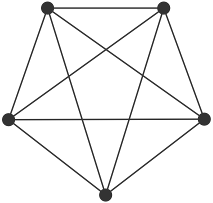
Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.



Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.



Beyond Sparsity

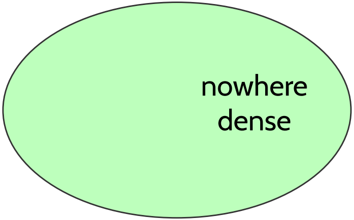
Goal: Find tractable classes beyond sparsity.

Beyond Sparsity

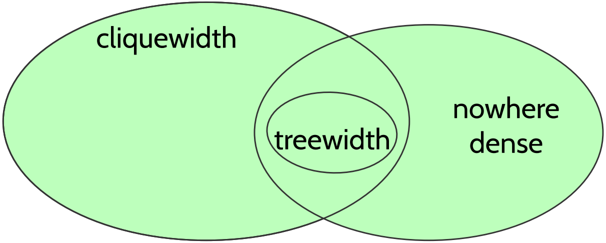
Goal: Find tractable classes beyond sparsity.

Today, we discuss some results, as well as possible candidates.

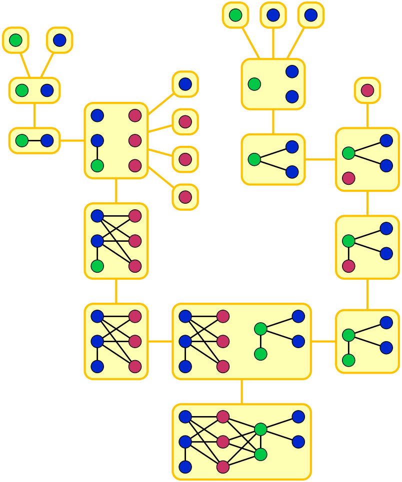
Classes with FPT first-order model-checking?



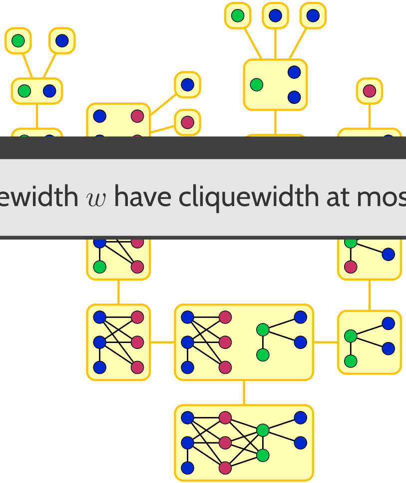
Classes with FPT first-order model-checking?



Cliquewidth

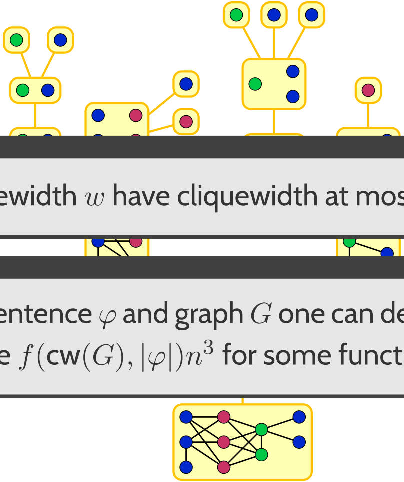


Cliquewidth



Graphs of treewidth w have cliquewidth at most $3 \cdot 2^{w-1}$.

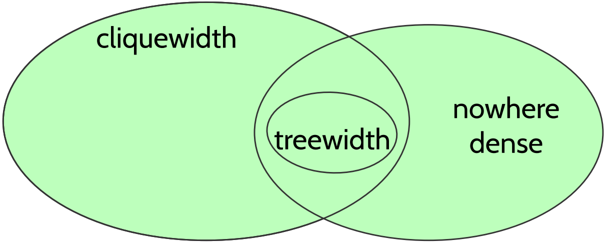
Cliquewidth



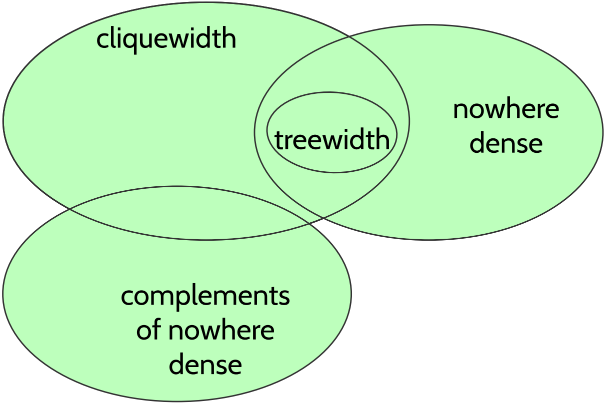
Graphs of treewidth w have cliquewidth at most $3 \cdot 2^{w-1}$.

For a MSO_1 sentence φ and graph G one can decide whether $G \models \varphi$ in time $f(\mathsf{cw}(G), |\varphi|)n^3$ for some function f.

Classes with FPT first-order model-checking?

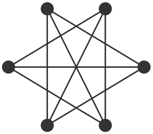


Classes with FPT first-order model-checking?



Complements

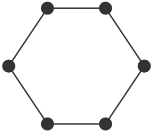
First-order model-checking is fpt on complements of nowhere dense classes by reduction.

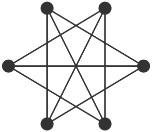


$\bar{G} \models \varphi$

Complements

First-order model-checking is fpt on complements of nowhere dense classes by reduction.

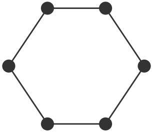




$$G \models \bar{\varphi} \iff \bar{G} \models \varphi$$

Complements

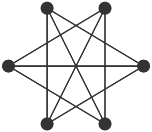
First-order model-checking is fpt on complements of nowhere dense classes by reduction.



$G \models \bar{\varphi}$

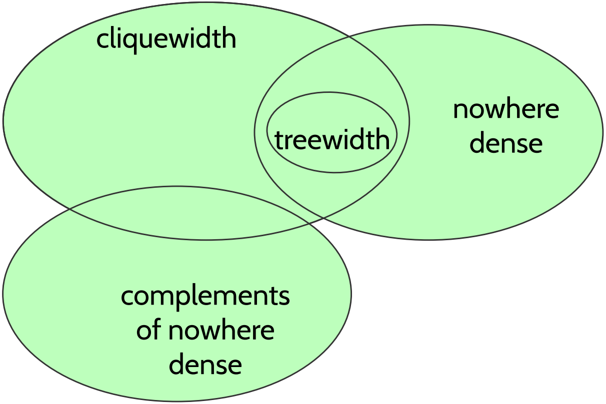
obtain $\bar{\varphi}$ from φ by swapping $\sim$ and $\not\sim$

$\Leftrightarrow$



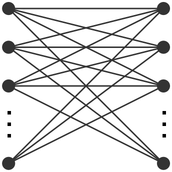
$\bar{G} \models \varphi$

Classes with FPT first-order model-checking?



Fully Bipartite Graphs

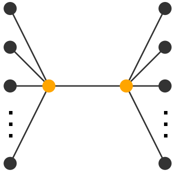
Can we do fpt model-checking on the class of fully bipartite graphs?



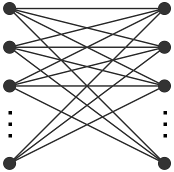
$G \models \varphi$

Fully Bipartite Graphs

Can we do fpt model-checking on the class of fully bipartite graphs?



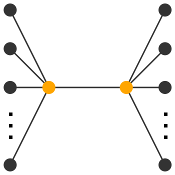
$G' \models \varphi'$



$G \models \varphi$

Fully Bipartite Graphs

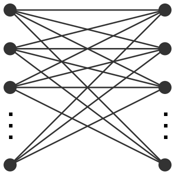
Can we do fpt model-checking on the class of fully bipartite graphs?



$G' \models \varphi'$

obtain φ' from φ by replacing $x{\sim}y$ with $\text{dist}(x,y)=3$

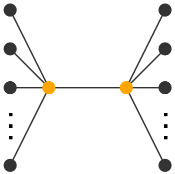
$\Leftrightarrow$



$G \models \varphi$

Fully Bipartite Graphs

Can we do fpt model-checking on the class of fully bipartite graphs?

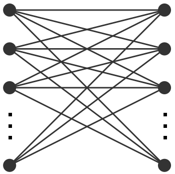


$G' \models \varphi'$

obtain φ' from φ by replacing $x{\sim}y$ with $\mathrm{dist}(x,y) = 3$

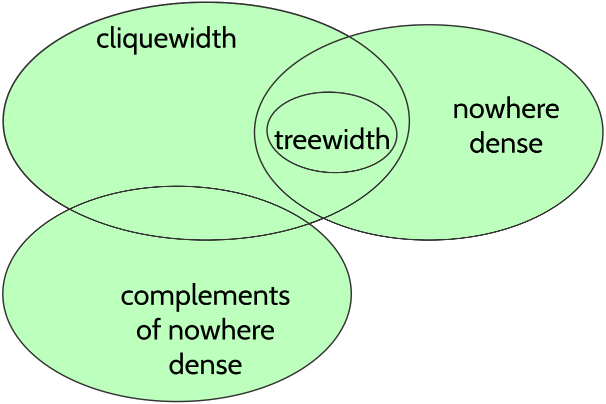
$\Leftrightarrow$

also restrict quantifiers to black vertices

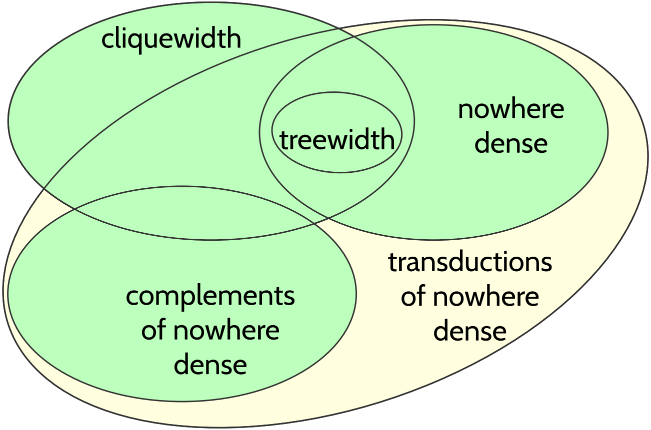


$G \models \varphi$

Classes with FPT first-order model-checking?



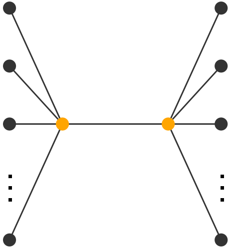
Classes with FPT first-order model-checking?



Interpretations

An interpretation $I = (\mu(x,y), \nu(x))$ maps G to $I(G)$.

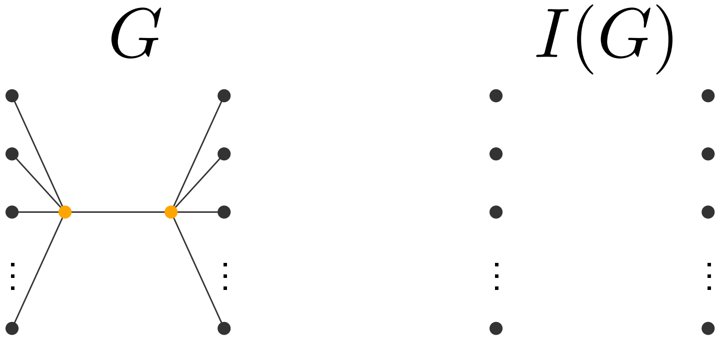
G $\qquad\qquad$ $I(G)$



Interpretations

An interpretation $I = (\mu(x, y), \nu(x))$ maps G to $I(G)$.

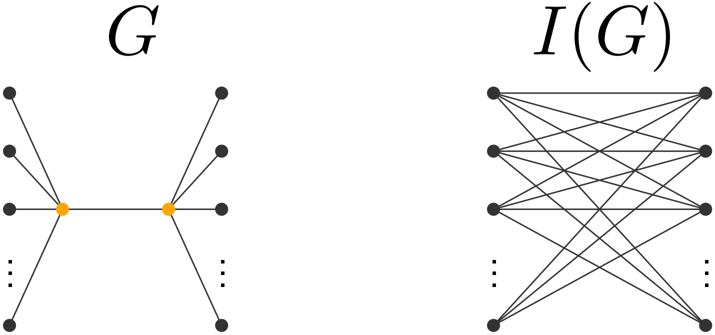
- vertices of $I(G)$: $\{v \colon G \models \nu(v)\}$.



Interpretations

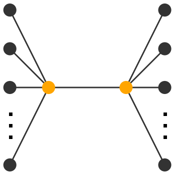
An interpretation $I = (\mu(x, y), \nu(x))$ maps G to $I(G)$.

- vertices of $I(G)$: $\{v \colon G \models \nu(v)\}$.
- edges of $I(G)$: $\{uv \colon G \models \mu(u, v)\}$.



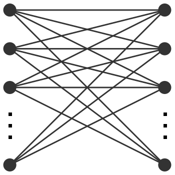
Interpretations

Assume we want to know whether $I(G) \models \varphi$.



$G \models \varphi'$

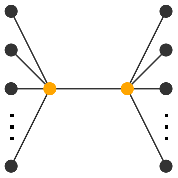
$$\Leftrightarrow$$



$I(G) \models \varphi$

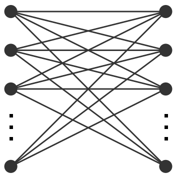
Interpretations

Assume we want to know whether $I(G) \models \varphi$. We can build a formula φ' from I and φ and instead evaluate $G \models \varphi'$.



$G \models \varphi'$

$$\Leftrightarrow$$

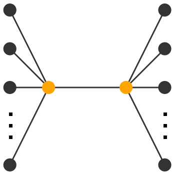


$I(G) \models \varphi$

Interpretations

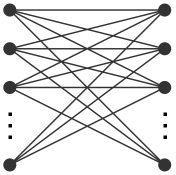
Assume we want to know whether $I(G) \models \varphi$. We can build a formula φ' from I and φ and instead evaluate $G \models \varphi'$.

Instead of asking whether $x \sim y$ in $I(G)$, ask whether $\mu(x, y)$ in G.



$G \models \varphi'$

$$\Leftrightarrow$$



$I(G) \models \varphi$

Transductions

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

Transductions

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees

Transductions

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees

- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$

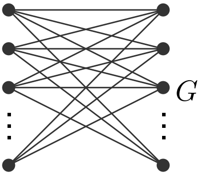
Transductions

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees

- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite G

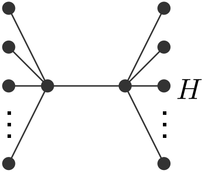


Transductions

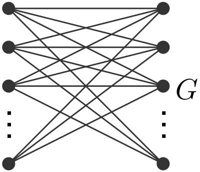
A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees



- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite G
- there is a tree H

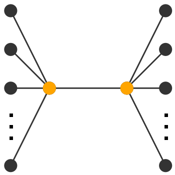


Transductions

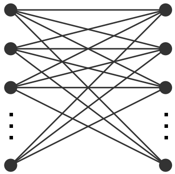
A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees



- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite G
- there is a tree H
- and a coloring H' of H



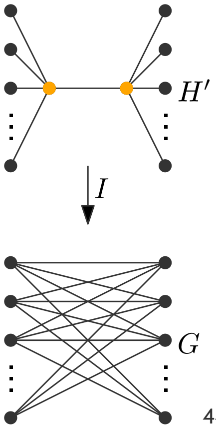
Transductions

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

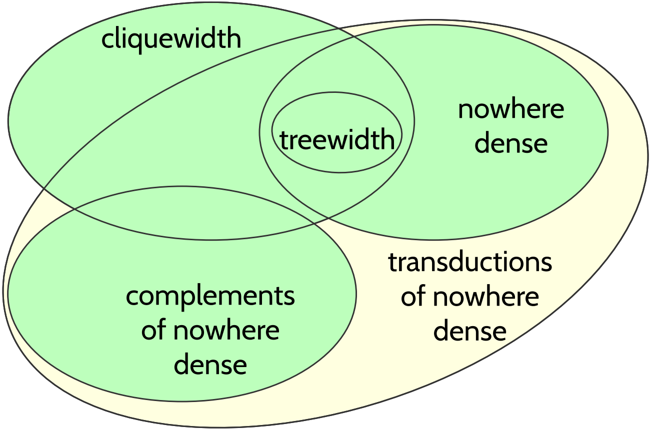
Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees

- There is $I = (\text{black}(x), \text{dist}(x,y) = 3)$
- such that if for every complete bipartite G
- there is a tree H
- and a coloring H' of H
- such that $G = I(H')$.



Classes with FPT first-order model-checking?



Transductions

We believe that model-checking is fpt on transductions of nowhere dense classes.

Transductions

We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don't know how to prove it.

Transductions

We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don't know how to prove it.

Assume we want to evalute $I(G) \models \varphi$ with G from a nowhere dense class.

Transductions

We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don't know how to prove it.

Assume we want to evalute $I(G) \models \varphi$ with G from a nowhere dense class. If we could construct G from $I(G)$, we could instead evaluate $G \models \varphi'$ (where φ' is obtained from I and φ).

Transductions

We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don't know how to prove it.

Assume we want to evalute $I(G) \models \varphi$ with G from a nowhere dense class. If we could construct G from $I(G)$, we could instead evaluate $G \models \varphi'$ (where φ' is obtained from I and φ).

For some sparse graph classes, this approach works.

Transductions

Gajarský, Hliněný, Lokshtanov, Obdržálek, Ramanujan 2018

Let $\mathcal{C}$ be a transduction of a class with bounded degree. First-order model-checking is fpt on $\mathcal{C}$.

Transductions

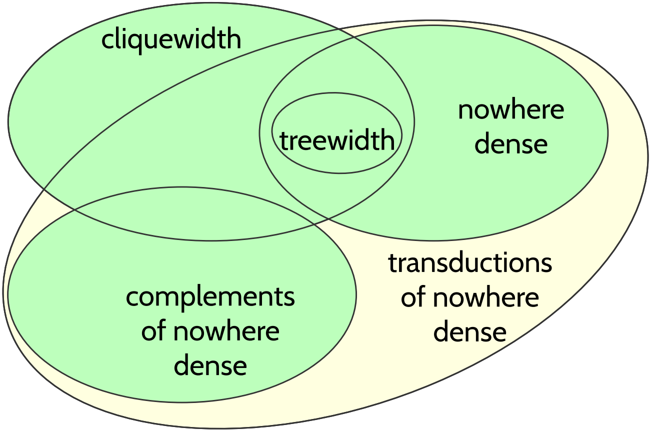
Gajarský, Hliněný, Lokshtanov, Obdržálek, Ramanujan 2018

Let $\mathcal{C}$ be a transduction of a class with bounded degree. First-order model-checking is fpt on $\mathcal{C}$.

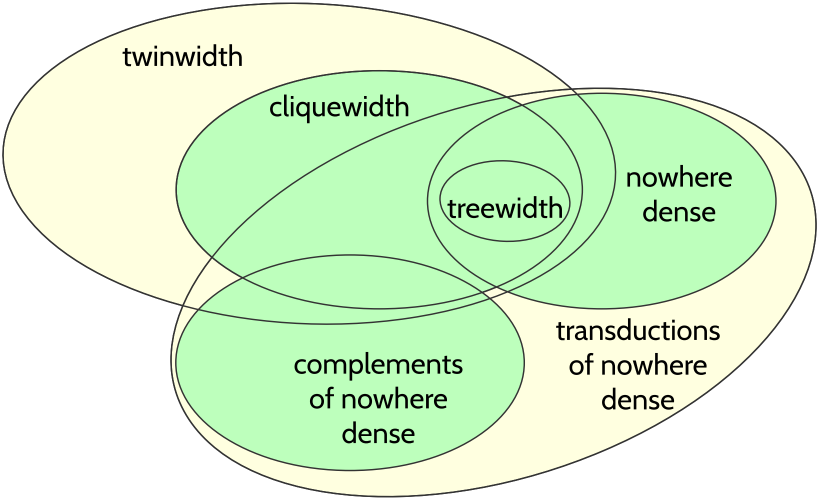
Bonnet, Dreier, Gajarský, Kreuzer, Mählmann, Toruńczyk 2022+

Let $\mathcal{C}$ be a transduction of a class with locally bounded treewidth. First-order model-checking is fpt on $\mathcal{C}$.

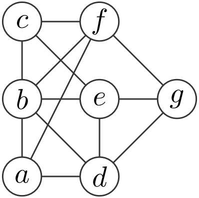
Classes with FPT first-order model-checking?



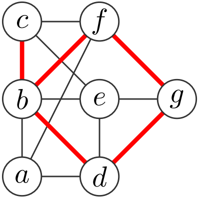
Classes with FPT first-order model-checking?



You already know normal graphs.

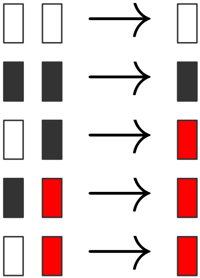


You already know normal graphs.

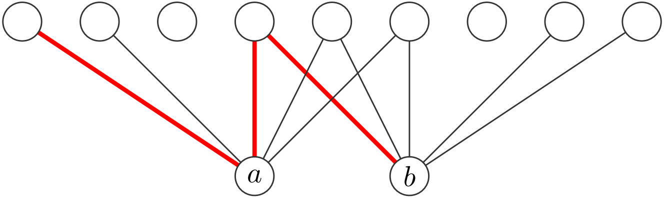


In *trigraphs* there are additional red error edges.

Contraction

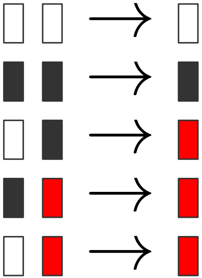


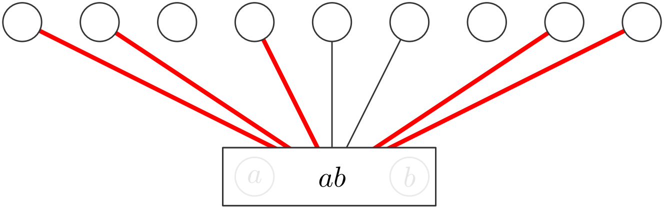
We can contract two (not neccessarily adjacent) vertices a and b. The edges of the new vertex ab follow this table.



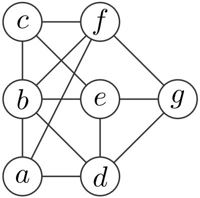
Contraction

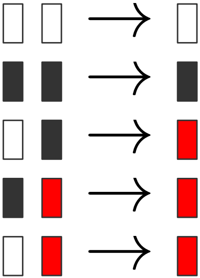
We can contract two (not neccessarily adjacent) vertices a and b. The edges of the new vertex ab follow this table.





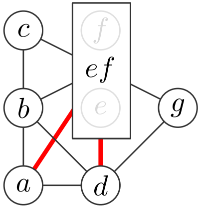
Contraction Sequences

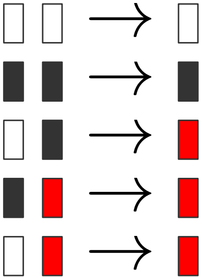




A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Contraction Sequences

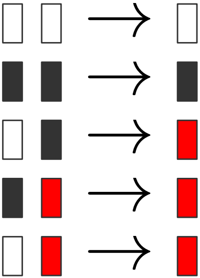


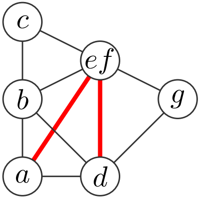


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

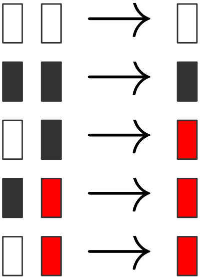


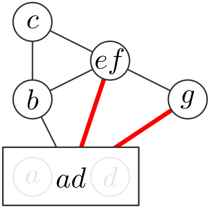


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

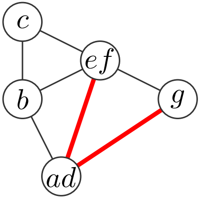


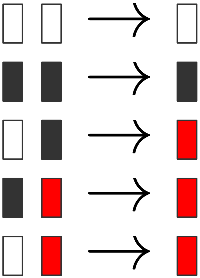


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

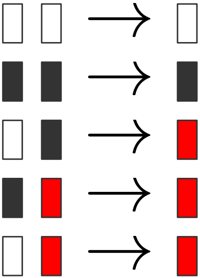


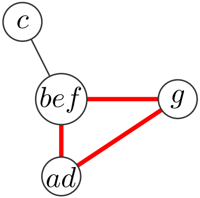


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

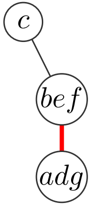


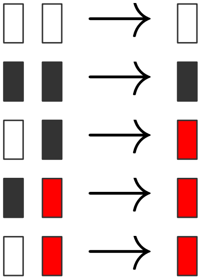


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

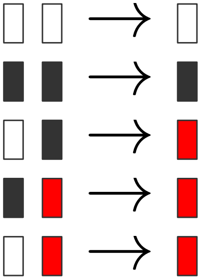


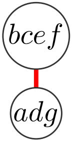


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

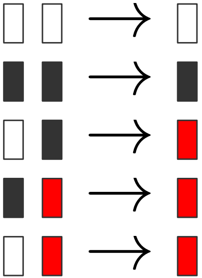




A *contraction sequence* is a sequence of contractions until only a single vertex is left.

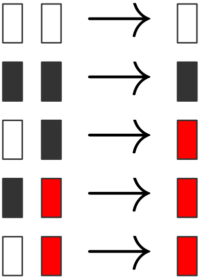
Example by Édouard Bonnet

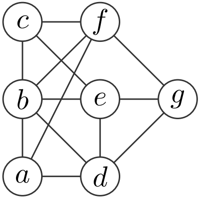
Contraction Sequences



A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Contraction Sequences

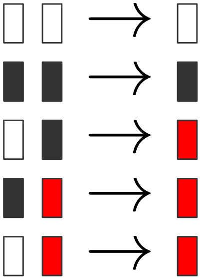


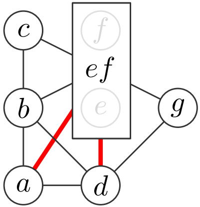


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

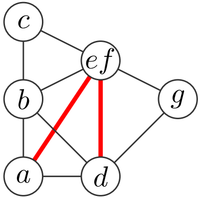


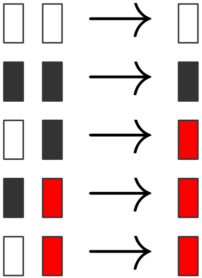


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

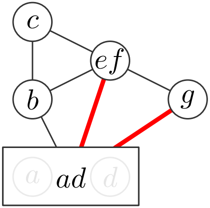


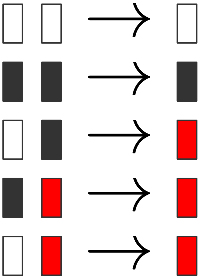


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

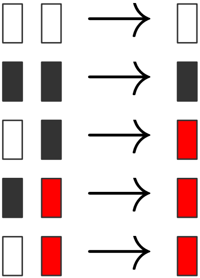


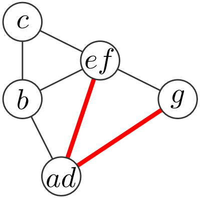


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

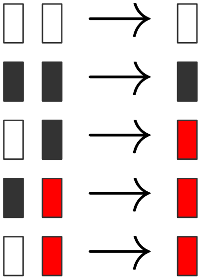


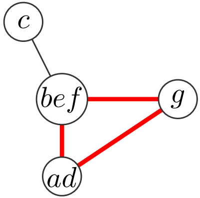


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

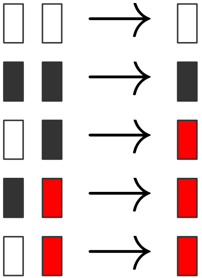


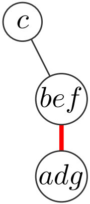


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

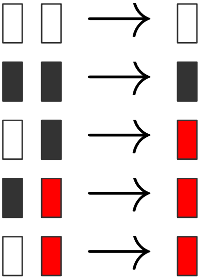


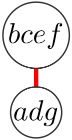


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

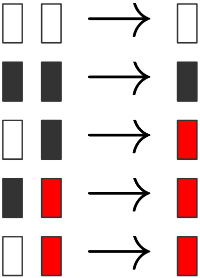


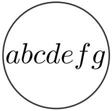


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences





Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Classes of Bounded Twinwidth

The following classes have bounded twinwidth

- planar graphs,
- classes with bounded cliquewidth.

Classes of Bounded Twinwidth

The following classes have bounded twinwidth

- planar graphs,
- classes with bounded cliquewidth.

The following classes do not have bounded twinwidth

- graphs with degree three.

First-Order Model-Checking

So far, nobody knows how to compute (approximate) contraction sequences of graphs with bounded twinwidth.

First-Order Model-Checking

So far, nobody knows how to compute (approximate) contraction sequences of graphs with bounded twinwidth. But once we do, model-checking is fpt.

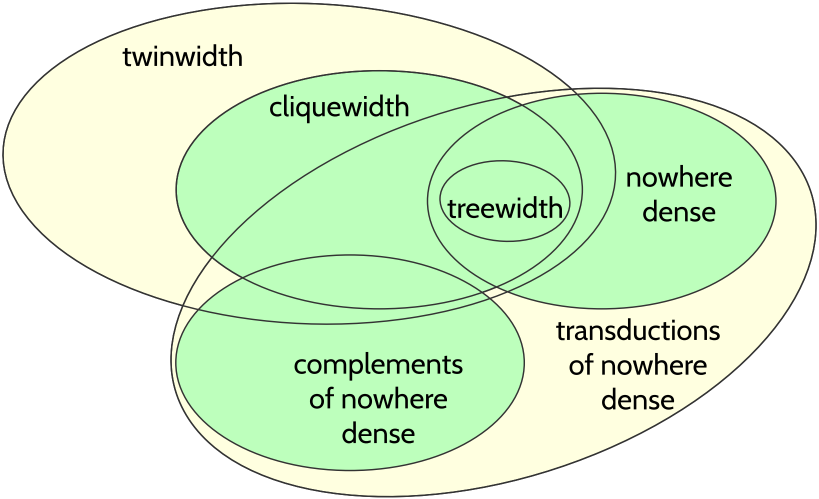
First-Order Model-Checking

So far, nobody knows how to compute (approximate) contraction sequences of graphs with bounded twinwidth. But once we do, model-checking is fpt.

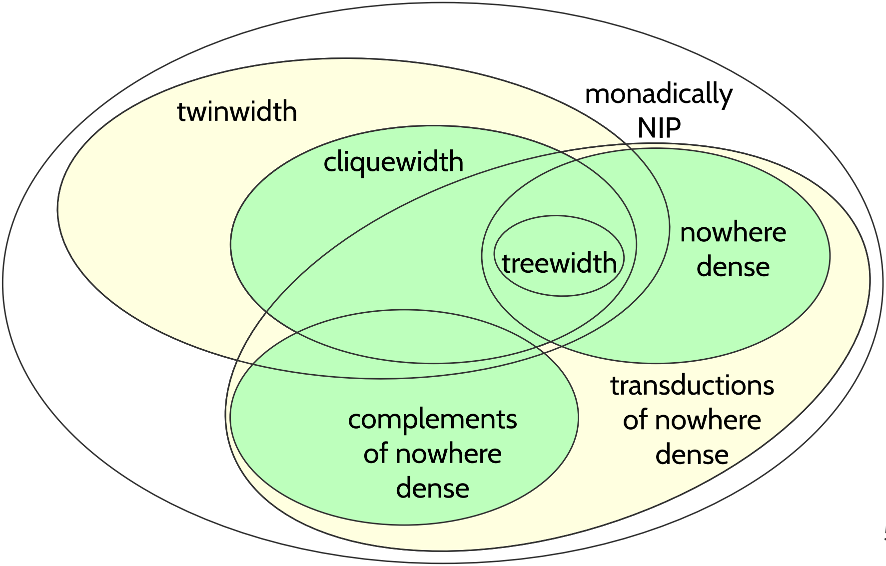
Bonnet, Kim, Thomassé, Watrigant 2021

Let $\mathcal{C}$ be a class of bounded twinwidth. Then first-order model-checking is fpt on $\mathcal{C}$, if one is additionally provided a contraction sequence of bounded twinwidth.

Classes with FPT first-order model-checking?

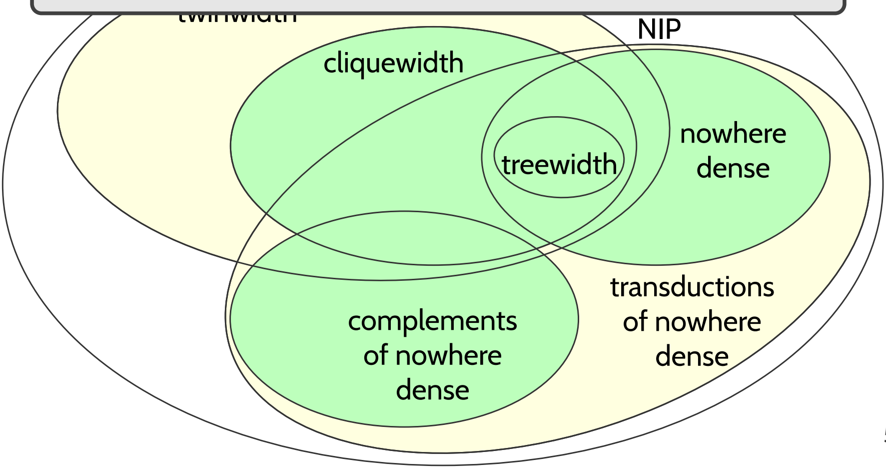


Classes with FPT first-order model-checking?



Classes with FPT first-order model-checking?

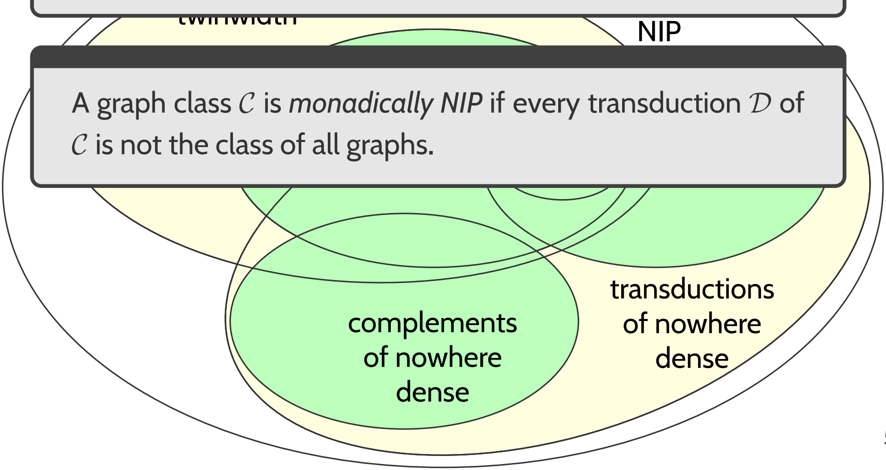
A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.



Classes with FPT first-order model-checking?

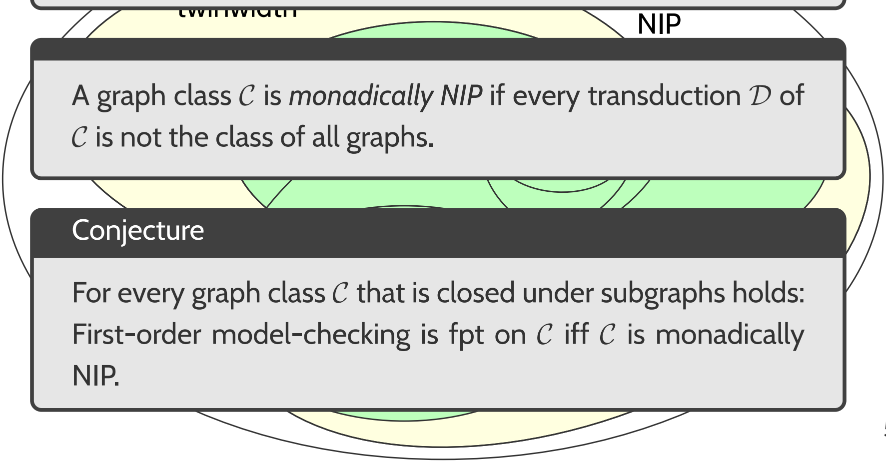
A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.

A graph class $\mathcal{C}$ is *monadically NIP* if every transduction $\mathcal{D}$ of $\mathcal{C}$ is not the class of all graphs.



Classes with FPT first-order model-checking?

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation I such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring H' of H such that $G = I(H')$.



A graph class $\mathcal{C}$ is *monadically NIP* if every transduction $\mathcal{D}$ of $\mathcal{C}$ is not the class of all graphs.

Conjecture

For every graph class $\mathcal{C}$ that is closed under subgraphs holds: First-order model-checking is fpt on $\mathcal{C}$ iff $\mathcal{C}$ is monadically NIP.

The End