Treewidth via Spined Categories

Zoltan A. Kocsis (CSIRO)

ACT 2021, Cambridge, UK

15 July 2021

joint work with **Benjamin Merlin Bumpus** (University of Glasgow)

Papers please

Z. A. K., Benjamin Merlin Bumpus:
Treewidth via Spined Categories
(this talk)
arXiv:2105.05372

Benjamin Merlin Bumpus, Z. A. K.:
Spined categories: generalizing tree-width beyond graphs
(journal article, submitted)
arXiv:2104.01841

slides: *www.existence.property/act21.pdf*

What we did (summary)

Context

Treewidth A numerical invariant defined on graphs.

Uses: Robertson-Seymour graph minor theorem.

Applications:

- **Courcelle's theorem**: every property of graphs definable in MSOL is linear time decidable on graphs of bounded treewidth.

Context

Treewidth Analogues

Fruitful research activity: define analogues of treewidth...

Context

Treewidth Analogues

Fruitful research activity: define analogues of treewidth...

- ... for hypergraphs and digraphs;

Context

Treewidth Analogues

Fruitful research activity: define analogues of treewidth...

- ... for hypergraphs and digraphs;
- ... for temporal graphs (edge sets change over time);

Context

Treewidth Analogues

Fruitful research activity: define analogues of treewidth...

- ... for hypergraphs and digraphs;
- ... for temporal graphs (edge sets change over time);
- ... and even fractional graphs.

The Problem

Obtaining treewidth analogues for other structures:
useful and *possible*.

[1] its use: dixit Wittgenstein

The Problem

Obtaining treewidth analogues for other structures:
useful and *possible.*

But ad-hoc. **We wanted:**

- A categorial description capturing its meaning[1]
- A uniform, categorial construction.

[1] its use: dixit Wittgenstein

Treewidth as Functor

We define

- **Spined categories**: categories with some extra structure.
- **Spined functors** preserve this extra structure.
- Examples: $\mathbf{Grph}_m$, $\mathbf{HGrph}_m$, posets ($\mathbf{Nat}$), etc.

Treewidth as Functor

We prove the following

Theorem
Given a spined category $\mathcal{C}$, either

- *There are no spined functors $F : \mathcal{C} \to \mathbf{Nat}$; or*
- *there is a distinguished functor (to be characterized later) $\Delta_{\mathcal{C}} : \mathcal{C} \to \mathbf{Nat}$.*

Moreover,

- *$\Delta_{\mathbf{Grph}_m}$ is treewidth,*
- *$\Delta_{\mathbf{HGrph}_m}$ is hypergraph treewidth,*

and so on.

Treewidth, briefly

Warning

Beware! By graph, we mean a combinatorist's graph:

- **Finite**
- **Irreflexive**
- **Undirected**
- **Without parallel edges**

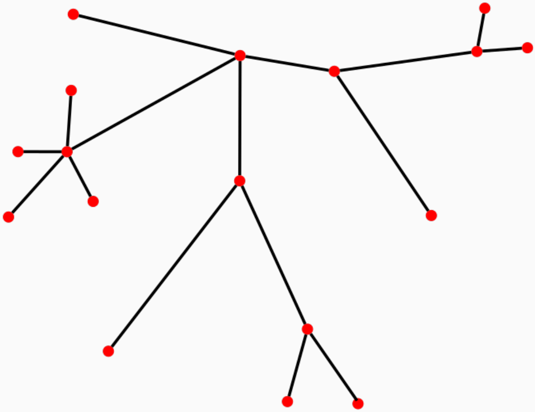
This clashes with the category theory convention. In particular, our category of graphs is not a quasitopos.

Treewidth: intuition

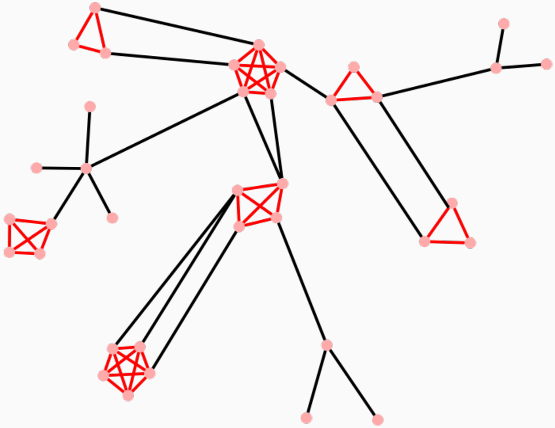
- **Treewidth**: a number $tw(G)$ describing each graph G.
- Captures how "tree-like" the *global* structure is.
- Trees are the graphs of treewidth 2. [2]
- Lower treewidth $\rightarrow$ more tree-like

[2] Cf. sets having h-level 2 in HoTT

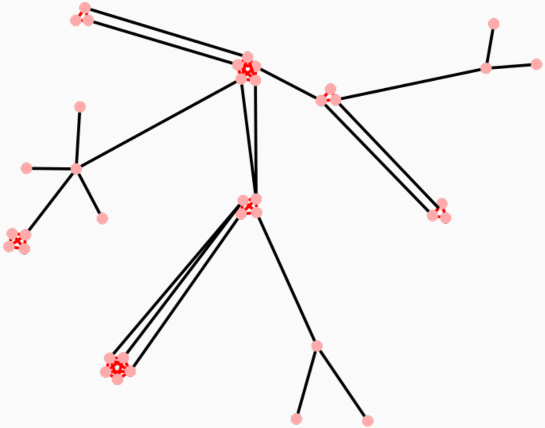
Example: Tree-like graphs



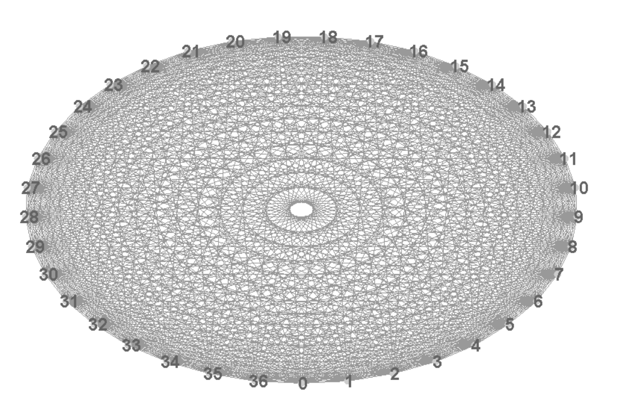
Example: Tree-like graphs



Example: Tree-like graphs



Compare: Complete graph on 37 vertices (treewidth 37)



The starting point

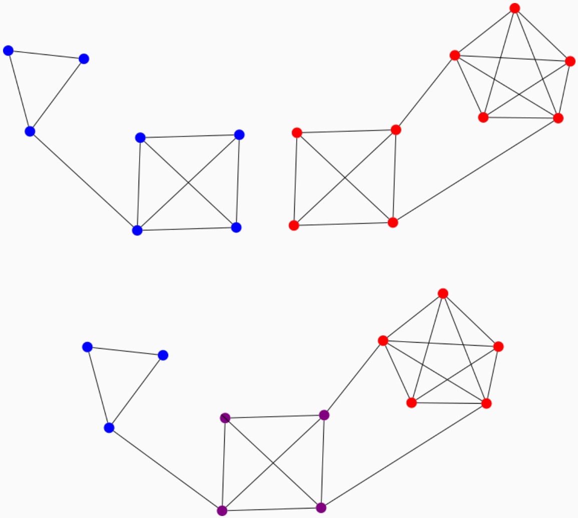


Figure 1: $\mathrm{tw}(B +_P R) = \max\{\mathrm{tw}(B), \mathrm{tw}(R)\}$

Spined Categories

Idea: Treewidth as Functor

Suggestive identity:

$$\mathrm{tw}(B +_P R) = \max\{\mathrm{tw}(B), \mathrm{tw}(R)\}$$

Idea: Treewidth as Functor

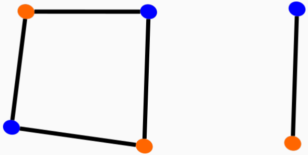
Suggestive identity:

$$\mathrm{tw}(B +_P R) = \max\{\mathrm{tw}(B), \mathrm{tw}(R)\}$$

Idea: tw as pushout-preserving functor $\mathbf{Grph} \to \mathbb{N}_{\leq}$

Issue 1: Homomorphisms

Issue: Graph homomorphisms do not preserve treewidth.



There is a graph homomorphism $C_4 \to C_2$, but we **don't** have

$$\mathrm{tw}(C_4) \leq \mathrm{tw}(C_2)$$

Solution 1

Observation: Graph monomorphisms do preserve treewidth. If $G \hookrightarrow H$, then $\mathrm{tw}(G) \leq \mathrm{tw}(H)$.

Naive solution: Consider the category $\mathbf{Grph}_m$ that has

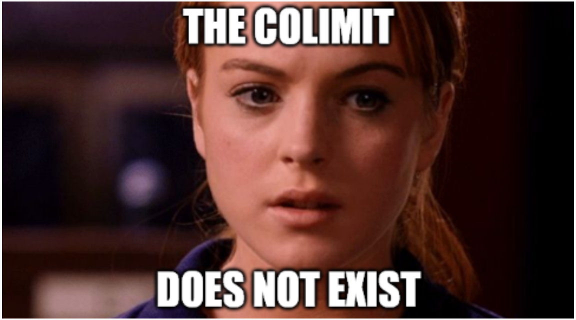
- Objects: simple graphs.
- Morphisms: monomorphisms of simple graphs.

and characterize tw as some kind of pushout-preserving functor

$$\mathrm{tw} : \mathbf{Grph}_m \to \mathbb{N}_{\leq}$$

Easy, right?!

Issue 2



Issue 2: No Pushouts

The category $\mathbf{Grph}_m$ lacks pushouts. Consider

$$\begin{tikzcd}
K_1 \arrow[r, hook] \arrow[d, hook] & K_2 \arrow[d, hook] \arrow[ddr, bend left] & \\
K_2 \arrow[r, hook] \arrow[drr, bend right] & K_2 +_1 K_2 & \\
& & K_2
\end{tikzcd}$$

where K_n is the complete graph on n vertices, i.e.
K_1 is $\bullet$ and K_2 is $\bullet \longrightarrow \bullet$.

Solution 2: Proxy Pushouts

$\mathbf{Grph}_m$ remembers *something* about the existence of pushouts in **Grph**

Proxy pushouts: the categorial ingredient, axiomatizes what $\mathbf{Grph}_m$ remembers.

Solution 2: Proxy Pushouts

$\mathbf{Grph}_m$ remembers *something* about the existence of pushouts in **Grph**

Proxy pushouts: the categorial ingredient, axiomatizes what $\mathbf{Grph}_m$ remembers.

Spine: the combinatorial ingredient, axiomatizes "complete" objects Ω_n: think "complete graphs".

Spined Categories I

Definition
A *spined category* consists of a category $\mathcal{C}$ equipped with the following additional structure:

- a sequence $\Omega : \mathbb{N} \to \operatorname{ob} \mathcal{C}$ called the *spine* of $\mathcal{C}$,

Spined Categories I

Definition

A *spined category* consists of a category $\mathcal{C}$ equipped with the following additional structure:

- a sequence $\Omega : \mathbb{N} \to \operatorname{ob} \mathcal{C}$ called the *spine* of $\mathcal{C}$,
- an operation $\mathfrak{P}$ (called the *proxy pushout*) that extends every diagram of the form
  $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ in $\mathcal{C}$

Spined Categories I

Definition

A *spined category* consists of a category $\mathcal{C}$ equipped with the following additional structure:

- a sequence $\Omega : \mathbb{N} \to \operatorname{ob}\mathcal{C}$ called the *spine* of $\mathcal{C}$,
- an operation $\mathfrak{P}$ (called the *proxy pushout*) that extends every diagram of the form

  $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ in $\mathcal{C}$

  to a distinguished commutative square

$$\begin{tikzcd} \Omega_n \arrow[r, "g"] \arrow[d, "h"'] & G \arrow[d, "\mathfrak{P}(g,h)_g"] \\ H \arrow[r, "\mathfrak{P}(g,h)_h"'] & \mathfrak{P}(g,h) \end{tikzcd}$$

  so that the following two conditions hold:

Spined Categories

Definition (cont.)
... so that the following two conditions hold:

SC1: If $X \in \operatorname{ob} \mathcal{C}$ we have $n \in \mathbb{N}$ such that $\mathcal{C}(X, \Omega_n) \neq \emptyset$.

Definition (cont.)
... so that the following two conditions hold:

SC1: If $X \in \operatorname{ob}\mathcal{C}$ we have $n \in \mathbb{N}$ such that $\mathcal{C}(X, \Omega_n) \neq \emptyset$.

SC2: Given any diagram of the form

$$\begin{tikzcd}
\Omega_n \arrow[r, "g"] \arrow[d, "h"'] & G \arrow[rr, "g'"] \arrow[d, "\mathfrak{P}(g,h)_g"] & & G' \arrow[dd, "\mathfrak{P}(g' \circ g, h' \circ h)_{g' \circ g}"] \\
H \arrow[r, "\mathfrak{P}(g,h)_h"'] \arrow[d, "h'"'] & \mathfrak{P}(g, h) & & \\
H' \arrow[rrr, "\mathfrak{P}(g' \circ g, h' \circ h)_{h' \circ h}"'] & & & \mathfrak{P}(g' \circ g, h' \circ h)
\end{tikzcd}$$

Definition (cont.)
... so that the following two conditions hold:

SC1: If $X \in \operatorname{ob} \mathcal{C}$ we have $n \in \mathbb{N}$ such that $\mathcal{C}(X, \Omega_n) \neq \emptyset$.

SC2: Given any diagram of the form

$$\begin{tikzcd}
\Omega_n \arrow[r, "g"] \arrow[d, "h"'] & G \arrow[rr, "g'"] \arrow[d, "\mathfrak{P}(g,h)_g"] & & G' \arrow[dd, "\mathfrak{P}(g'\circ g,h'\circ h)_{g'\circ g}"] \\
H \arrow[r, "\mathfrak{P}(g,h)_h"'] \arrow[d, "h'"'] & \mathfrak{P}(g,h) \arrow[drr, dashed, "{(g',h')}"] & & \\
H' \arrow[rrr, "\mathfrak{P}(g'\circ g,h'\circ h)_{h'\circ h}"'] & & & \mathfrak{P}(g' \circ g, h' \circ h)
\end{tikzcd}$$

$\exists!(g', h') : \mathfrak{P}(g, h) \to \mathfrak{P}(g' \circ g, h' \circ h)$ making it commute.

Spined Functor

The obvious notion of morphism between spined categories.

Definition
Consider spined categories $(\mathcal{C}, \Omega^{\mathcal{C}}, \mathfrak{P}^{\mathcal{C}})$ and $(\mathcal{D}, \Omega^{\mathcal{D}}, \mathfrak{P}^{\mathcal{D}})$. We call a functor $F : \mathcal{C} \to \mathcal{D}$ a *spined functor* if it

1. *preserves the spine*, i.e. $F \circ \Omega^{\mathcal{C}} = \Omega^{\mathcal{D}}$, and
2. *preserves proxy pushouts*, i.e. the F-image of every proxy pushout square in $\mathcal{C}$ is a proxy pushout square in $\mathcal{D}$.

One can state the latter equationally, by demanding that the equalities $F[\mathfrak{P}^{\mathcal{C}}(g, h)] = \mathfrak{P}^{\mathcal{D}}(Fg, Fh)$,
$F\mathfrak{P}^{\mathcal{C}}(g, h)_g = \mathfrak{P}^{\mathcal{D}}(Fg, Fh)_{Fg}$ and
$F\mathfrak{P}^{\mathcal{C}}(g, h)_h = \mathfrak{P}^{\mathcal{D}}(Fg, Fh)_{Fh}$ all hold.

Examples

The poset $\mathbb{N}_{\leq}$ regarded as a category, with

Spine: $\Omega_n = n$
Proxy pushouts: pushouts (i.e. suprema)

We denote this spined category **Nat**. It will play an important role as the codomain of our ”abstract treewidth”!

Examples

The category $\mathbf{Grph}_m$ (simple graphs and monomorphisms), with

Spine: $\Omega_n = K_n$, the complete graph on n vertices

Proxy pushouts: the proxy pushout

$$\begin{tikzcd} \Omega_n \arrow[r, "g"] \arrow[d, "h"'] & G \arrow[d, "\mathfrak{P}(g,h)_g"] \\ H \arrow[r, "\mathfrak{P}(g,h)_h"'] & \mathfrak{P}(g,h) \end{tikzcd}$$

is just the pushout square in **Grph**.

Similarly for $\mathbf{HGrph}_m$ (hypergraphs and monomorphisms).

Other Examples

- The category $\mathbf{FinSet}_m$ (sets and monomorphisms) with Ω_n denoting the n-element set, and proxy pushouts as in **Set**.
- The poset $\mathbb{N}_{div}$, with least common multiples as proxy pushouts,

$$\Omega_n = \prod_{p \leq n} p^n$$

  where p ranges over the primes.
- Many other combinatorial examples...

Treewidth as Functor

Easy Observations

The map that sends each graph to the size of its largest complete subgraph is a spined functor $\omega : \mathbf{Grph}_m \to \mathbf{Nat}$.

From here on we focus on spined categories $\mathcal{C}$ such that there exists at least one $s : \mathcal{C} \to \mathbf{Nat}$.

Triangulation Functor

We define a distinguished S-functor $\Delta_{\mathcal{C}} : \mathcal{C} \to \mathbf{Nat}$ on each category $\mathcal{C}$ with some $s : \mathcal{C} \to \mathbf{Nat}$. This will...

- ... be canonical, and constructed uniformly.
- ... satisfy a *maximality* property.
- ... coincide with treewidth when $\mathcal{C} = \mathbf{Grph}_m$.

Pseudo-chordal Objects

Definition
Take an object $X \in \operatorname{ob} \mathcal{C}$. We call X *pseudo-chordal* if for any two spined functors $F, G : \mathcal{C} \to \mathbf{Nat}$, we have

$$F[X] = G[X].$$

I.e. if all treewidth-like functors agree on X.

Pseudo-chordal Objects

Definition
Take an object $X \in \operatorname{ob} \mathcal{C}$. We call X *pseudo-chordal* if for any two spined functors $F, G : \mathcal{C} \to \mathbf{Nat}$, we have

$$F[X] = G[X].$$

I.e. if all treewidth-like functors agree on X.

We "know the treewidth" of pseudo-chordal objects X:

$$\Delta_{\mathcal{C}}[X] = s[X].$$

Triangulation Functor

We can use pseudo-chordal objects as "test objects" to define $\Delta_{\mathcal{C}}$ on all other objects.

Definition
We define the *triangulation functor* $\Delta_{\mathcal{C}} : \mathcal{C} \to \mathbf{Nat}$ via

$$\Delta_{\mathcal{C}}[X] = \inf \{\Delta_{\mathcal{C}}[H] \mid \exists f \colon X \to H \text{ s.t. } H \text{ is pseudo-chordal}\}$$

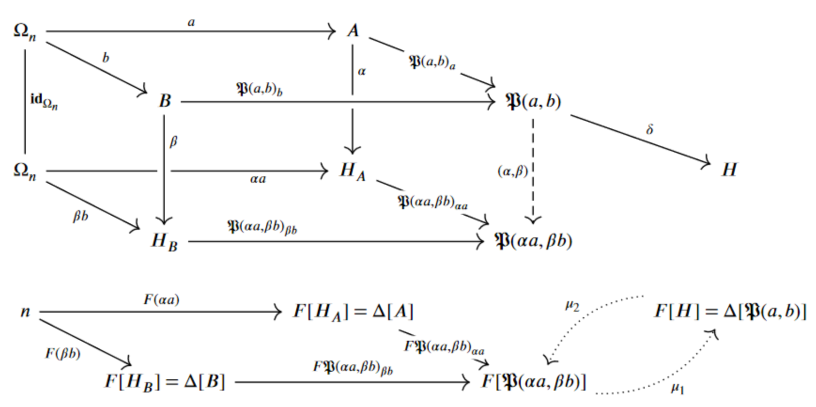
for each $X \in \operatorname{ob} \mathcal{C}$.

Main Theorem

Theorem
The triangulation functor $\Delta_{\mathcal{C}} : \mathcal{C} \to \mathbf{Nat}$ *is*

- *a functor* $\mathcal{C} \to \mathbf{Nat}$.
- *a spined functor on* $\mathcal{C}$.
- ***the** object-wise maximal spined functor* $\mathcal{C} \to \mathbf{Nat}$.

Just stare at the following diagram ;)



Treewidth as Functor, Finally

Theorem

1. $\Delta_{\mathbf{Grph}_m}$ *coincides with treewidth.*
2. $\Delta_{\mathbf{HGrph}_m}$ *coincides with hypergraph treewidth.*
3. *A similar category of modular graphs yields modular treewidth.*

Computing $\Delta_{\mathcal{C}}$

Consider a spined category $(\mathcal{C}, \Omega, \mathfrak{P})$ such that

1. All Hom-sets $\mathcal{C}(X, Y)$ are finite and enumerable;
2. Equality of morphisms is decidable and $\circ$ is computable;
3. Proxy pushouts $\mathfrak{P}(g, h)$ are computable;
4. $\mathcal{C}$ has finitely many objects over Ω_n (up to iso).

There is a uniform (but slooow) algorithm that computes $\Delta_{\mathcal{C}}$ that works in any such category $\mathcal{C}$.

https://github.com/zaklogician/act2021-code

Conclusion

Payoff

We get "abstract tree decompositions": we can write algorithms for objects of bounded $\Delta_{\mathcal{C}}$ just like we do for graphs of bounded treewidth.

Future work

- Work out more specific examples.
- Dualize! "subgraphs :: treewidth" as "colorings :: ???".
- Relation with Baez and Courser's Structured Cospans?
- Aspiration: a categorial Courcelle's Theorem

Thanks!

Questions?

Reprise: Why not...

- **Adhesive categories?** What goes wrong? Posets are never adhesive, so we would not have a codomain for Δ.
- **Algebraic and order-theoretic examples?** Seemingly difficult. By dualizing, we might have something for finitely presented groups, but details have to be worked out.
- **Algebraic issues?** Pushouts arise from free products in algebraic theories. These tend to be infinite. But when not (e.g. bounded join-semilattices), you need to choose a spine carefully to avoid measurability issues.
- **Spatial, topological examples?** I'm very hopeful (but note that finite topology is order theory).

Reprise: Glossary

- **Robertson-Seymour theorem**: the "set" of undirected graphs, when partially ordered by the graph minor relation, is well-quasi-ordered. (E.g. Wagner's forbidden minors, K_5 and $K_{3,3}$ as obstructions to planarity!)
- **Kruskal's tree theorem**: Robertson-Seymour for trees. Much easier to prove.
- **Courcelle's theorem**: Every graph property definable in MSO is decidable in linear time on graphs of bounded treewidth.

Reprise: Treewidth Definition 1

Treewidth has many equivalent definitions.

- **Most useful:** via tree decompositions.
- **Most relevant:** via chordal completion.
- The latter is easier to understand.

Reprise: Treewidth Definition 2

Definition
A graph G is *chordal* if every cycle $C \subseteq G$ (of length > 3) has a *chord*: an edge of G connecting two non-consecutive vertices of C

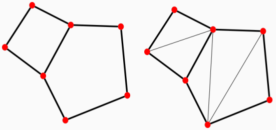
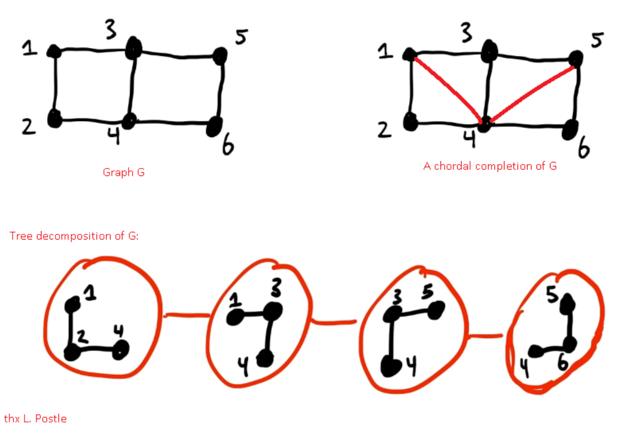


Figure 2: The graph on the left is not chordal. The graph on the right is chordal.

Reprise: Treewidth Definition 3

Definition
Given $G \hookrightarrow H$ such that H is chordal, we say that H is a *chordal completion* of G. The ***treewidth*** of G is the size of the largest complete graph that occurs (as a subgraph) in every chordal completion of G.

Reprise: Treewidth Example



Reprise: Treewidth as Functor: Proof*

Theorem

$\Delta_{\mathbf{Grph}_m}$ *coincides with treewidth.*

Proof.

1. *"Size of largest complete subgraph" is an S-functor* $\omega : \mathbf{Grph}_m \to \mathbf{Nat}$.
2. *If* X *has a pseudo-chordal completion* Y*, then it also has a chordal completion* Y' *with* $\omega(Y) = \omega(Y')$ *(just take the chordal completion of* Y*).*
3. $\mathrm{tw}(X)$ *is the size of the largest complete subgraph that occurs in every chordal completion of* X*, so we're done.*

$\square$

Computing $\Delta_{\mathcal{C}}$: Idea*

- Pseudo-chordal objects are hard to find: you need to know all S-functors to begin with.
- But main theorem relies on two properties: Ω_n is pseudo-chordal, and pseudo-chordal objects are closed under proxy pushouts.

Computing $\Delta_{\mathcal{C}}$: Idea*

- Pseudo-chordal objects are hard to find: you need to know all S-functors to begin with.
- But main theorem relies on two properties: Ω_n is pseudo-chordal, and pseudo-chordal objects are closed under proxy pushouts.
- Pseudo-chordals form the largest set with these two properties!

Computing $\Delta_{\mathcal{C}}$: Idea*

- Pseudo-chordal objects are hard to find: you need to know all S-functors to begin with.
- But main theorem relies on two properties: Ω_n is pseudo-chordal, and pseudo-chordal objects are closed under proxy pushouts.
- Pseudo-chordals form the largest set with these two properties!
- Using computational assumptions, we can construct the *smallest set* with these two properties inductively!

Computing $\Delta_{\mathcal{C}}$: Idea*

- Pseudo-chordal objects are hard to find: you need to know all S-functors to begin with.
- But main theorem relies on two properties: Ω_n is pseudo-chordal, and pseudo-chordal objects are closed under proxy pushouts.
- Pseudo-chordals form the largest set with these two properties!
- Using computational assumptions, we can construct the *smallest set* with these two properties inductively!
- This yields a (slooow) algorithm to compute $\Delta_{\mathcal{C}}$.

Reprise: Measurability Proofs*

- Spined categories interact nicely via spined functors.
- E.g. spined functors reflect measurability.
- $\mathbf{HGrph}_m$ **is measurable** via the Gaifman functor $G : \mathbf{HGrph}_m \to \mathbf{Grph}_m$ + existence of $\Delta_{\mathbf{Grph}_m}$
- $\mathbf{FinSet}_m$ **is not**: via the functor that forgets edges $V : \mathbf{Grph}_m \to \mathbf{FinSet}_m$ + maximality of $\Delta_{\mathbf{Grph}_m}$

Reprise: Measurability Proofs*

- Spined categories interact nicely via spined functors.
- If $\mathcal{C}$ is measurable, and there is $F : \mathcal{D} \to \mathcal{C}$, then $\mathcal{D}$ is measurable.
- $\mathbf{HGrph}_m$ is measurable: the Gaifman functor $\mathbf{HGrph}_m \to \mathbf{Grph}_m$ that sends each hypergraph to its graph skeleton is spined.
- $\mathbf{FinSet}_m$ is not measurable: the functor that forgets edges, $\mathbf{Grph}_m \to \mathbf{FinSet}_m$ is spined. But generally $\mathrm{tw}(X) \ngeq |V(X)|$.

An open question

Consider the category which has

Objects: finite posets
Morphisms: order embeddings

equipped with the usual pushout construction.

Is there a spine which turns this into a measurable spined category?