Minimal classes of graphs of unbounded tree-width and clique-width

Dan Cocks

This talk is based on joint work with Robert Brignall

23rd November 2022

Agenda

1. Motivation - algorithmic problems on graphs
2. Types of graph class
3. Tree-width in minor-closed classes
4. Clique-width in dense hereditary classes
5. Tree-width and clique-width in sparse hereditary classes

"The formulation of the problem is often far more essential than its solution."
Albert Einstein

Algorithmic problems on graphs

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

"The formulation of the problem is often far more essential than its solution."
Albert Einstein

decision problem defined on a graph - a problem that can be posed as a yes-no question based on the input of a graph.

Algorithmic problems on graphs

"The formulation of the problem is often far more essential than its solution."
Albert Einstein

decision problem defined on a graph - a problem that can be posed as a yes-no question based on the input of a graph.

algorithm - a process or set of rules to be followed in calculations or other problem-solving operations.

Algorithmic problems on graphs

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

"The formulation of the problem is often far more essential than its solution."
Albert Einstein

decision problem defined on a graph - a problem that can be posed as a yes-no question based on the input of a graph.

algorithm - a process or set of rules to be followed in calculations or other problem-solving operations.

Example: The seven bridges of Konigsberg. Is there a walk through the city that crosses each bridge once and only once? [Euler 1736]

Decision problems defined on a graph

The Open University

Question

For a decision problem defined on a graph which is generally a 'hard' problem, are there types of graph for which the problem becomes 'easy'?

Decision problems defined on a graph

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Question

For a decision problem defined on a graph which is generally a 'hard' problem, are there types of graph for which the problem becomes 'easy'?

Only considering problems definable in MSO_1 and MSO_2 logic:

Theorem (Courcelle, 1990)

For every graph class of bounded tree-width, every problem definable in MSO_2 logic can be solved in time linear in the number of vertices of the graph.

Theorem (Courcelle, Makowsky, Rotics, 2000)

For every graph class of bounded clique-width, every problem definable in MSO_1 logic can be solved in time linear in the number of vertices of the graph.

(MSO_1 logic more restricted than MSO_2 logic)

Examples: Problem defined on a graph

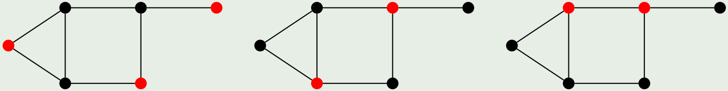
Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Examples of problems defined in MSO_2/ MSO_1 logic:

- MSO_2 Hamiltonicity
- MSO_1 Isomorphism problem
- MSO_1 Dominating set problem...

A dominating set for a graph $G = (V, E)$ is a subset D of the vertices V such that every vertex not in D is adjacent to at least one member of D.

Example (Dominating sets)



Decision problem

Does graph G contain a dominating set of size at most k?

Challenge

To characterise those graphs at the boundary where hard problems become easy.

Challenge

To characterise those graphs at the boundary where hard problems become easy.

or specifically

Challenge

To characterise those graphs at the boundary where hard problems become easy.

or specifically

....to identify those graph classes that are obstructions to bounded tree-width or clique-width (′minimal classes′).

2. Types of graph class

Graph containment operations I

Graph classes

Graphs are undirected and simple (no loops, or multiple edges).
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Graph containment operations I

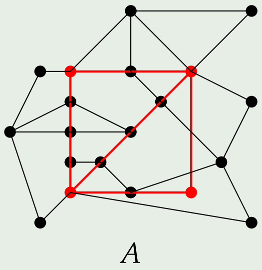
Graph classes

Graphs are undirected and simple (no loops, or multiple edges).

Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Vertex deletions, edge deletions, suppressions)

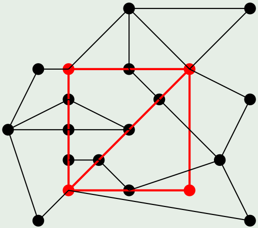


Graph containment operations I

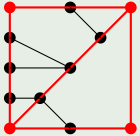
Graphs are undirected and simple (no loops, or multiple edges).
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Vertex deletions, edge deletions, suppressions)



A



B

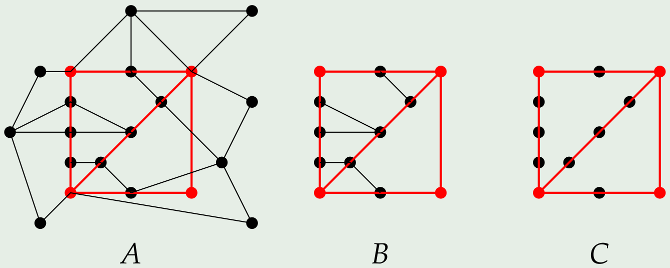
Graph containment operations I

Graph classes

Graphs are undirected and simple (no loops, or multiple edges).
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Vertex deletions, edge deletions, suppressions)



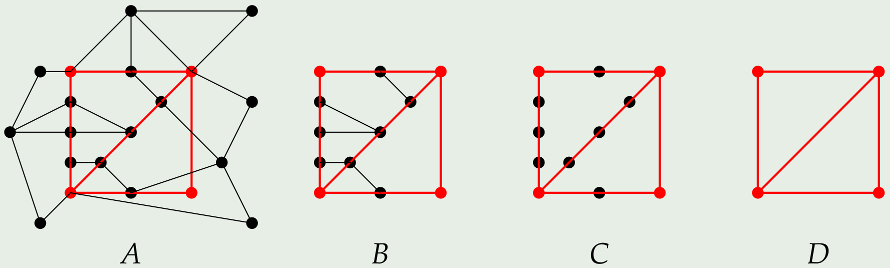
Graph containment operations I

Graph classes

Graphs are undirected and simple (no loops, or multiple edges).
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Vertex deletions, edge deletions, suppressions)



Graph containment operations II

Graph classes

Graphs are undirected and simple (no loops, or multiple edges).

Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

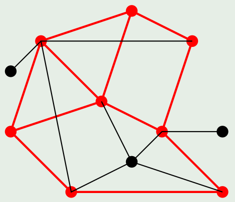
Graph containment operations II

Graph classes

Graphs are undirected and simple (no loops, or multiple edges).
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Contracting edges)



A

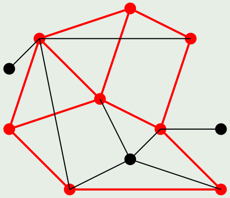
Graph containment operations II

Graphs are undirected and simple (no loops, or multiple edges).

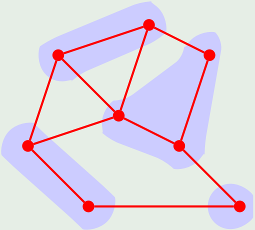
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Contracting edges)



A



B

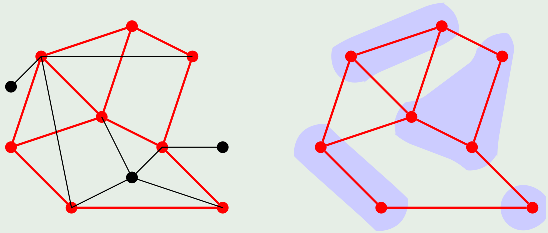
Graph containment operations II

Graph classes

Graphs are undirected and simple (no loops, or multiple edges).
Four possible graph operations considered:

- Removing a vertex
- Removing an edge
- Suppressing a deg. 2 vertex (opposite to subdivision)
- Contracting an edge

Example (Contracting edges)



A B D

Graph ordering and graph classes

Graph classes

Relation	Operations	Downsets	Examples
Induced subgraph ($H \leq_I G$)	vertex removal	Hereditary class	Forests, planar, complete, complete bipartite, bounded degree, line graphs
Minor ($H \leq_M G$)	vertex removal, edge removal, contraction	Minor closed class	Forests, planar

- Hereditary graph class $\mathcal{C}$: if $G \in \mathcal{C}$ and $H \leq_I G$ then $H \in \mathcal{C}$.
- Minor-closed graph class $\mathcal{C}$: if $G \in \mathcal{C}$ and $H \leq_M G$ then $H \in \mathcal{C}$.

Classes defined by forbidden graphs

- A collection of graphs that is hereditary or minor-closed is completely determined once we know a set of minimal elements that are not in the set. We call these a minimal forbidden set.

Classes defined by forbidden graphs

- A collection of graphs that is hereditary or minor-closed is completely determined once we know a set of minimal elements that are not in the set. We call these a minimal forbidden set.

Theorem (Wagner 1937)

A graph is planar if and only if it contains neither K_5 nor $K_{3,3}$ as a minor.

3. Tree-width in minor-closed classes

Origins of tree-width

- 1950s: For trees, NP-hard problems are often easy [using recursion and dynamic programming algorithms].

Origins of tree-width

- 1950s: For trees, NP-hard problems are often easy [using recursion and dynamic programming algorithms].
- 1960/70s: Extend to "Series-parallel" graphs.

Origins of tree-width

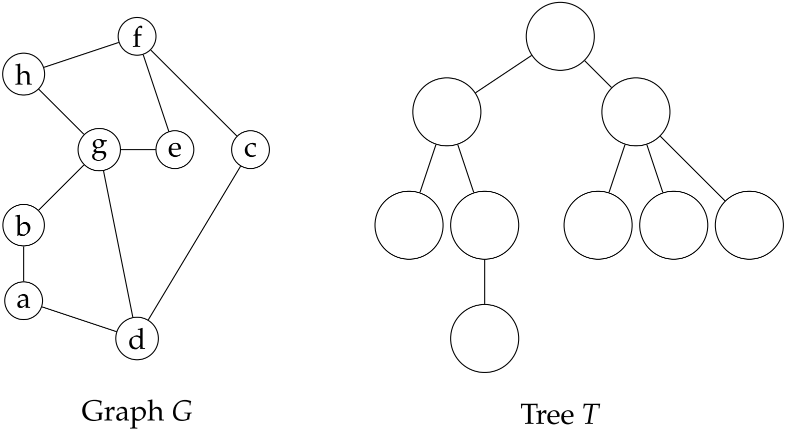
- 1950s: For trees, NP-hard problems are often easy [using recursion and dynamic programming algorithms].
- 1960/70s: Extend to "Series-parallel" graphs.
- 1970/80s: Tree-width born. A parameter that measures how 'tree-like' a graph is ['Discovered' several times by Bertelè & Brioschi (1972), Halin (1976) and Robertson & Seymour (1984)].

Origins of tree-width

- 1950s: For trees, NP-hard problems are often easy [using recursion and dynamic programming algorithms].
- 1960/70s: Extend to "Series-parallel" graphs.
- 1970/80s: Tree-width born. A parameter that measures how ’tree-like’ a graph is [’Discovered’ several times by Bertelè & Brioschi (1972), Halin (1976) and Robertson & Seymour (1984)].

Tree-decomposition example

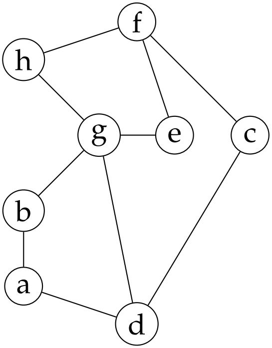
Tree-width



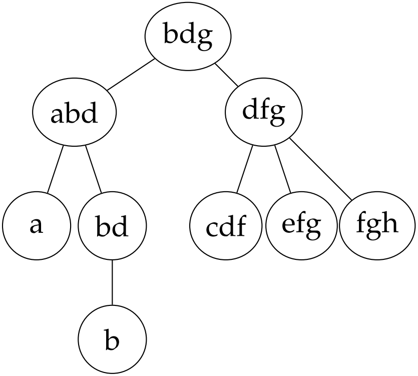
Objective: Put the vertices of graph G into the bags of tree T.

Tree-decomposition example

Tree-width



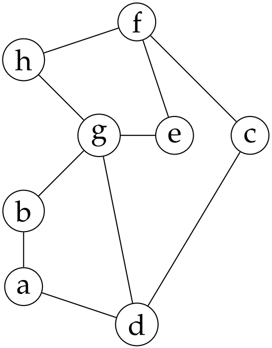
Graph G



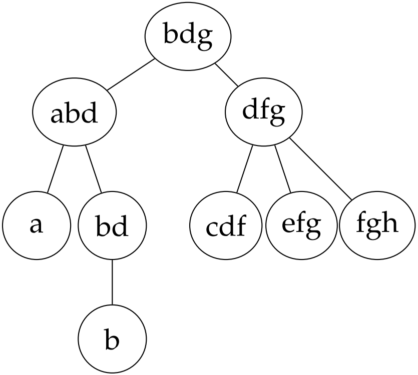
Tree T

Rule 1 : Every vertex of G must go into at least one bag.

Tree-decomposition example



Graph G

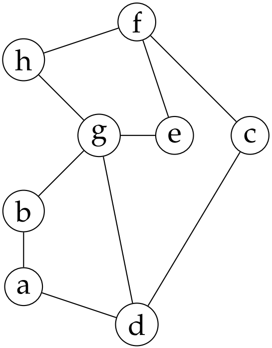


Tree T

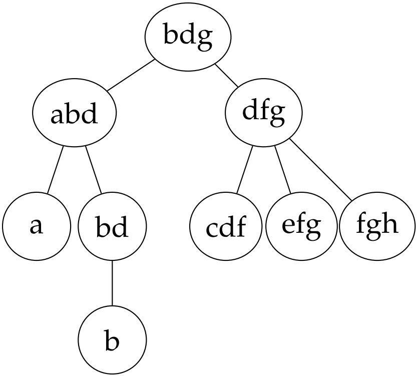
Rule 2 : The endvertices of every edge of G must be together in some bag.

Tree-decomposition example

Tree-width



Graph G

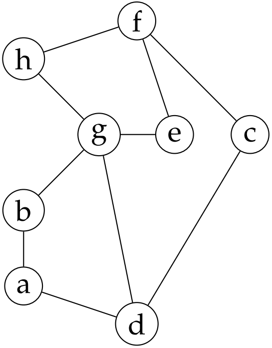


Tree T

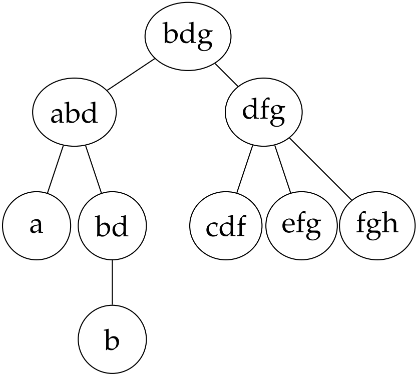
Rule 3 : For every vertex v of G, the bags containing v must be connected in T.

Tree-decomposition example

Tree-width



Graph G



Tree T

The width of a tree-decomposition is the maximum bag size minus 1.

Tree-width definition

The tree-width $tw(G)$ of G is the least width of any tree-decomposition of G.

Tree-width definition

The tree-width $tw(G)$ of G is the least width of any tree-decomposition of G.

- Trees have tree-width 1.
- Series-parallel graphs have tree-width 2.
- K_n has tree-width $n - 1$.
- A graph with 'large enough' average degree contains K_n as a minor.
- If a class $\mathcal{C}$ contains graphs of unbounded average degree (dense) then it has unbounded tree-width.

Neil Robertson and Paul Seymour
The Graph Minor Theorem

Figure: Neil Robertson

Figure: Paul Seymour

Diestel: 'The Graph Minor Theorem dwarfs any other result in graph theory.... and can be counted among the deepest theorems that mathematics has to offer.'

Robertson and Seymour - Graph Minors

1. Excluding a forest (1983)
2. Algorithmic aspects of tree-width (1986)
3. Planar tree-width (1984)
4. Tree-width and well-quasi-ordering (1990)
5. Excluding a planar graph (1986)
6. Disjoint paths across a disc (1986)
7. Disjoint paths on a surface (1988)
8. A Kuratowski theorem for general surfaces (1990)
9. Disjoint crossed paths (1990)
10. Obstructions to tree-decomposition (1991)
11. Circuits on a surface (1994)
12. Distance on a surface (1995)
13. The disjoint paths problem (1995)
14. Extending an embedding (1995)
15. Giant steps (1996)
16. Excluding a non-planar graph (2003)
17. Taming a vortex (1999)
18. Tree-decompositions and well-quasi-ordering (2003)
19. Well-quasi-ordering on a surface (2004)
20. Wagner's Conjecture (2004)
21. Graphs with unique linkages (2009)
22. Irrelevant vertices in linkage problems (2012)
23. Nash-Williams' immersion conjecture (2010)

Robertson and Seymour results

Graph Minors

Theorem (Grid Theorem)

For every integer r there is an integer k such that every graph of tree-width at least k has an $r \times r$ grid minor.

Robertson and Seymour results

Theorem (Grid Theorem)

For every integer r there is an integer k such that every graph of tree-width at least k has an $r \times r$ grid minor.

Theorem

Given a graph H, the graphs without an H minor have bounded tree-width if and only if H is planar.

Robertson and Seymour results

Graph Minors

Theorem (Grid Theorem)

For every integer r there is an integer k such that every graph of tree-width at least k has an $r \times r$ grid minor.

Theorem

Given a graph H, the graphs without an H minor have bounded tree-width if and only if H is planar.

Theorem (Graph Minor Theorem)

Every minor-closed graph class can be defined by a finite set of forbidden minors.

Robertson and Seymour results

Graph Minors

Theorem (Grid Theorem)

For every integer r there is an integer k such that every graph of tree-width at least k has an $r \times r$ grid minor.

Theorem

Given a graph H, the graphs without an H minor have bounded tree-width if and only if H is planar.

Theorem (Graph Minor Theorem)

Every minor-closed graph class can be defined by a finite set of forbidden minors.

Theorem

For every surface S there exists a finite set of graphs $H_1, \ldots, H_n$ such that a graph is embeddable in S if and only if it contains none of $H_1, \ldots, H_n$ as a minor.

Recap of key points - Tree-width

- In bounded tree-width graph classes many problems which are generally hard become 'easy' (i.e. an algorithm exists)
- Tree-width measures how 'tree-like' a graph is
- Graph classes with 'large' tree-width contain a 'large' grid minor.
- For minor-closed graph classes, a class without a graph H minor has bounded tree-width if and only if H is planar.
- If a class $\mathcal{C}$ contains graphs of unbounded average degree (dense) then it has unbounded tree-width.

4. Clique-width in dense hereditary classes

Origins of clique-width

- 1990s: Introduced by Courcelle, Engelfreit and Rozenberg.
- A generalisation of tree-width that can deal with dense graphs.
- Also MIM-width, MM-width, module-width, NLC-width, path-width, rank-width and others.

Clique-width - Building a graph

Clique-width - Building a graph

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Set of labels Σ. You have 4 operations to build a labelled graph:

1. Create a new vertex with a label $i \in \Sigma$.
2. Join all vertices labelled i to all labelled j, where $i, j \in \Sigma$, $i \neq j$.
3. Relabel every vertex labelled i with j.
4. Disjoint union of two previously-constructed graphs.

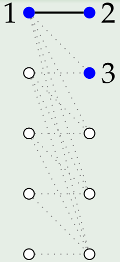
Clique-width - Building a graph

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Set of labels Σ. You have 4 operations to build a labelled graph:

1. Create a new vertex with a label $i \in \Sigma$.
2. Join all vertices labelled i to all labelled j, where $i, j \in \Sigma$, $i \neq j$.
3. Relabel every vertex labelled i with j.
4. Disjoint union of two previously-constructed graphs.

Example (Building a chain graph)



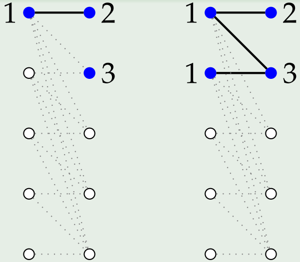
Clique-width - Building a graph

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Set of labels Σ. You have 4 operations to build a labelled graph:

1. Create a new vertex with a label $i \in \Sigma$.
2. Join all vertices labelled i to all labelled j, where $i, j \in \Sigma$, $i \neq j$.
3. Relabel every vertex labelled i with j.
4. Disjoint union of two previously-constructed graphs.

Example (Building a chain graph)



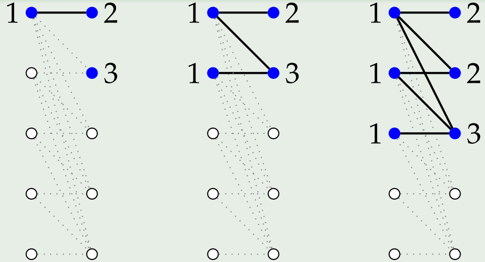
Clique-width - Building a graph

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Set of labels Σ. You have 4 operations to build a labelled graph:

1. Create a new vertex with a label $i \in \Sigma$.
2. Join all vertices labelled i to all labelled j, where $i, j \in \Sigma$, $i \neq j$.
3. Relabel every vertex labelled i with j.
4. Disjoint union of two previously-constructed graphs.

Example (Building a chain graph)



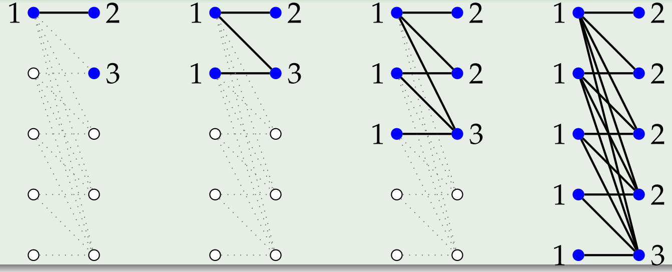
Clique-width - Building a graph

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Set of labels Σ. You have 4 operations to build a labelled graph:

1. Create a new vertex with a label $i \in \Sigma$.
2. Join all vertices labelled i to all labelled j, where $i, j \in \Sigma$, $i \neq j$.
3. Relabel every vertex labelled i with j.
4. Disjoint union of two previously-constructed graphs.

Example (Building a chain graph)



Clique-width - Building a graph

Set of labels Σ. You have 4 operations to build a labelled graph:

1. Create a new vertex with a label $i \in \Sigma$.
2. Join all vertices labelled i to all labelled j, where $i, j \in \Sigma$, $i \neq j$.
3. Relabel every vertex labelled i with j.
4. Disjoint union of two previously-constructed graphs.

- Clique-width, $cw(G) =$ size of smallest Σ needed to build G.
- Clique-width of a class $\mathcal{C}$

$$cw(\mathcal{C}) = \max_{G \in \mathcal{C}} cw(G)$$

if this exists.

Tree-width v clique-width

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Graph class	**Tree-width**	**Clique-width**
Forest	1	≤ 3
$(n \times n)$-grid ($n \geq 3$)	n	$n+1$
K_n	$n-1$	≤ 3

Tree-width v clique-width

Graph class	**Tree-width**	**Clique-width**
Forest	1	≤ 3
$(n \times n)$-grid $(n \geq 3)$	n	$n+1$
K_n	$n-1$	≤ 3

$$H \leq_M G \Rightarrow tw(H) \leq tw(G)$$

$$H \leq_I G \Rightarrow cw(H) \leq cw(G)$$

$$cw(G) \leq 2^{tw(G)+1} + 1$$

Tree-width v clique-width

Graph class	**Tree-width**	**Clique-width**
Forest	1	≤ 3
$(n \times n)$-grid ($n \geq 3$)	n	$n+1$
K_n	$n-1$	≤ 3

$$H \leq_M G \Rightarrow tw(H) \leq tw(G)$$

$$H \leq_I G \Rightarrow cw(H) \leq cw(G)$$

$$cw(G) \leq 2^{tw(G)+1} + 1$$

- There are more decision problems that becomes easy on graphs of bounded tree-width than become easy on graphs of bounded clique-width.
- There are more classes of bounded clique-width than bounded tree-width.

What is a minimal class?

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

- A minimal class: a hereditary class of graphs of unbounded clique-width that has no proper subclass of unbounded clique-width.
- A proper subclass : Hereditary class $\mathcal{D}$ is a proper subclass of hereditary class $\mathcal{C}$ if there exists a non-trivial forbidden graph F which is in $\mathcal{C}$ but not in $\mathcal{D}$.

Minimal classes found prior to 2020

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

- Bipartite permutations and unit interval graphs [Lozin, 2011]
- Split permutation graphs and bichain graphs [Atminas, Brignall, Lozin, Stacho, 2015]
- A countably infinite collection of minimal classes, built on an infinite grid and an infinite binary word α (to be explained) [Collins, Foniok, Korpelainen, Lozin, Zamaraev, 2017]

Why set out to build a framework for dense hereditary graph classes?

Why set out to build a framework for dense hereditary graph classes?

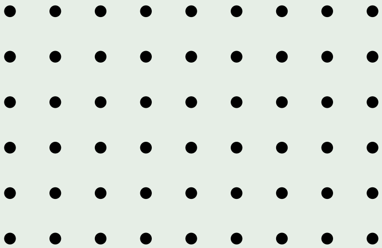
- To include all minimal classes in one model.
- To identify many new such classes that fit the same model.
- To identify the characteristics that minimal classes have in common.

A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

- Built on an infinite grid of vertices $\mathfrak{P}$ creating an infinite graph $\mathfrak{P}^{\delta}$.
- $\delta = (\alpha, \beta, \gamma)$ is a triple of objects that define the edges between consecutive columns, edges between non-consecutive columns (called bonds), and edges within columns.
- The hereditary graph class $\mathfrak{G}^{\delta}$ is the set of all finite induced subgraphs of $\mathfrak{P}^{\delta}$.

Example (Vertex grid $\mathfrak{P}$)

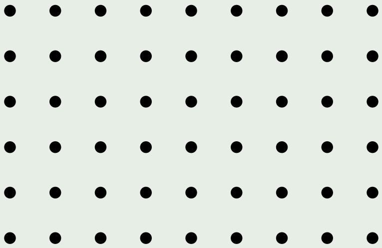


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123 \cdots$)

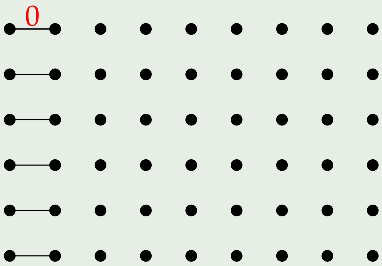


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123\cdots$)

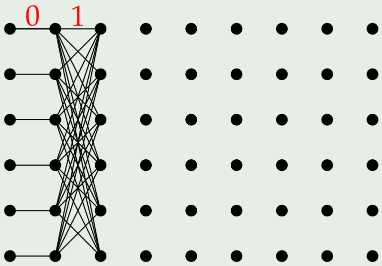


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

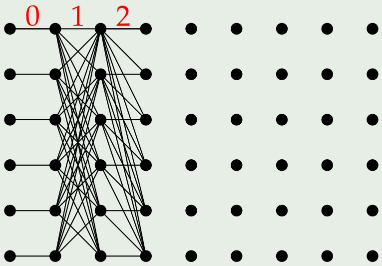
Example ($\alpha = 01230123\cdots$)



A framework for dense hereditary graph classes

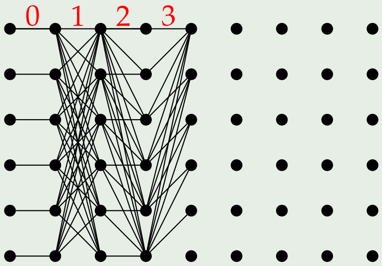
α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123\cdots$)



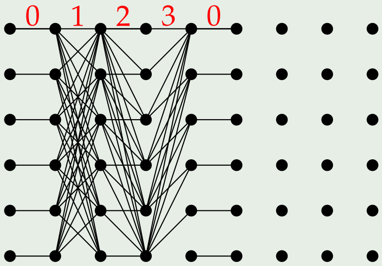
α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123\cdots$)



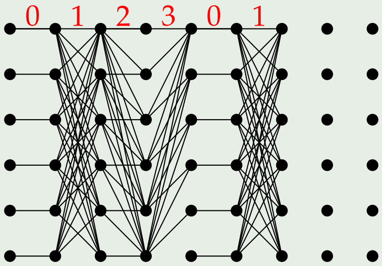
α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123\cdots$)



α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

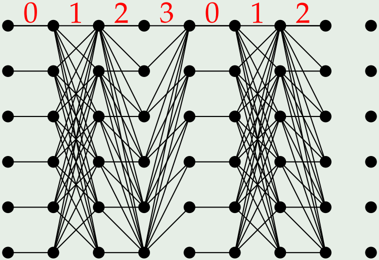
Example ($\alpha = 01230123\cdots$)



A framework for dense hereditary graph classes

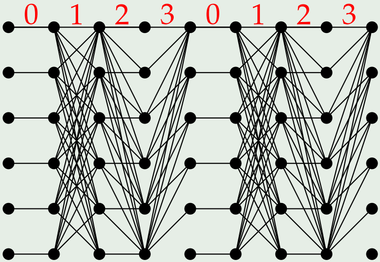
α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123\cdots$)



α is an infinite word from the alphabet $\{0, 1, 2, 3\}$ that defines edges between consecutive columns of the grid.
[A matching (0), the complement of a matching (1), a chain (2) and the complement of a chain (3)].

Example ($\alpha = 01230123\cdots$)



A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

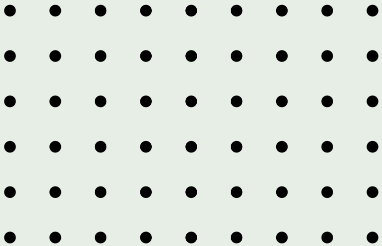
β is a symmetric subset of pairs of natural numbers (x, y). If $(x, y) \in \beta$ then every vertex in column x is adjacent to every vertex in column y. These edges between non-consecutive columns are called bonds.

A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

β is a symmetric subset of pairs of natural numbers (x, y). If $(x, y) \in \beta$ then every vertex in column x is adjacent to every vertex in column y. These edges between non-consecutive columns are called bonds.

Example ($\beta = \{(1,3),(4,9)\cdots\}$)

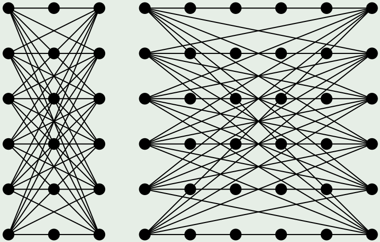


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

β is a symmetric subset of pairs of natural numbers (x, y). If $(x, y) \in \beta$ then every vertex in column x is adjacent to every vertex in column y. These edges between non-consecutive columns are called bonds.

Example ($\beta = \{(1,3),(4,9)\cdots\}$)

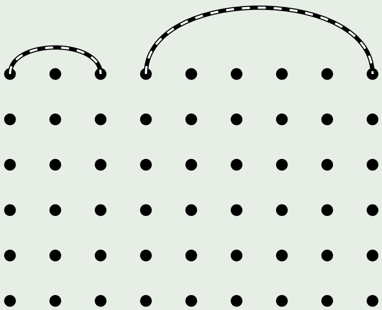


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

β is a symmetric subset of pairs of natural numbers (x, y). If $(x, y) \in \beta$ then every vertex in column x is adjacent to every vertex in column y. These edges between non-consecutive columns are called bonds.

Example ($\beta = \{(1,3),(4,9)\cdots\}$)



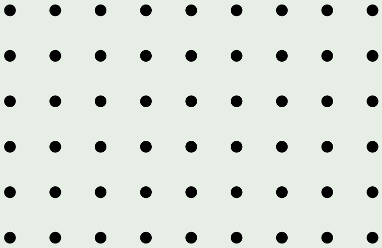
γ is an infinite binary word. If the j-th letter of γ is 0 then vertices in column j form an independent set and if it is 1 they form a clique.

A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

γ is an infinite binary word. If the j-th letter of γ is 0 then vertices in column j form an independent set and if it is 1 they form a clique.

Example ($\gamma = 010001001\cdots$)

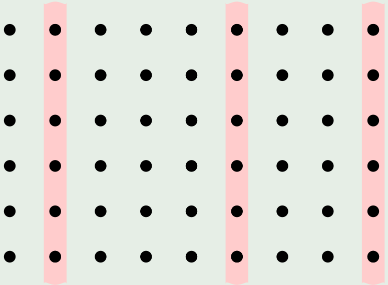


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

γ is an infinite binary word. If the j-th letter of γ is 0 then vertices in column j form an independent set and if it is 1 they form a clique.

Example ($\gamma = 010001001\cdots$)

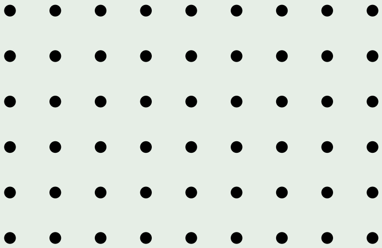


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

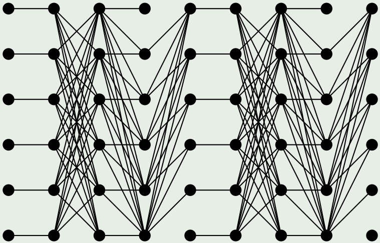
Combining α, β and γ edges . . .

Example ($\delta = (\alpha, \beta, \gamma)$)



Combining α, β and γ edges . . .

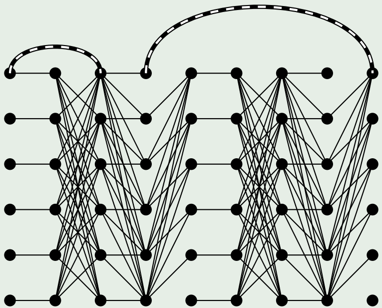
Example ($\delta = (\alpha, \beta, \gamma)$)



A framework for dense hereditary graph classes

Combining α, β and γ edges . . .

Example ($\delta = (\alpha, \beta, \gamma)$)

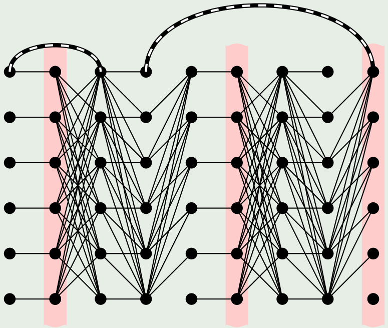


A framework for dense hereditary graph classes

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Combining α, β and γ edges …

Example ($\delta = (\alpha, \beta, \gamma)$)



When does $\mathfrak{G}^\delta$ have unbounded clique-width?

Approach to proof:

- Choose a sequence of graphs $\{G_n\} \in \mathfrak{G}^\delta$, typically an $(n \times n)$ square grid.
- Use Ramsey theory to prove there must exist a stage in construction of G_n such that the number of labels required to complete construction is greater than some $f(n) \to \infty$ as $n \to \infty$ so that clique-width is unbounded.

Let $\mathcal{N}^\delta$ be the number of different neighbourhoods between two rows of $\mathfrak{P}^\delta$.

Theorem (Brignall and C., 2022)

For any triple $\delta = (\alpha, \beta, \gamma)$ the hereditary class $\mathfrak{G}^\delta$ has unbounded clique-width if and only if $\mathcal{N}^\delta$ is unbounded.

When is $\mathfrak{G}^\delta$ a minimal class?

The Open University

For an infinite word ω:

- A factor of ω is a finite sequence of consecutive letters ω^*.
- ω is recurrent if each of its factors occurs in it infinitely many times.
- ω is almost periodic if for each factor ω^* of ω there exists a constant $\mathfrak{L}(\omega^*)$ such that every factor of ω of length at least $\mathfrak{L}(\omega^*)$ contains ω^* as a factor.
- Periodic $\Rightarrow$ almost periodic $\Rightarrow$ recurrent.

The same terms can be applied to δ-factors of δ-triples.

When is $\mathfrak{G}^\delta$ a minimal class?

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

We define a parameter $\mathcal{M}^\beta$ determined only by the bond set β that measures the number of distinct neighbourhoods between two disjoint intervals of a single row of $\mathfrak{P}^\delta$.

Theorem (Brignall and C., 2022)

A hereditary graph class $\mathfrak{G}^\delta$ that has unbounded clique-width is also minimal if

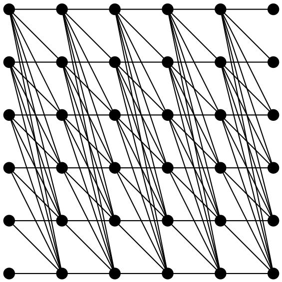
1. *δ is recurrent,*
2. *for any factor δ^* of δ, the subgraphs induced on the columns between two consecutive copies of δ^* (the δ-factor 'gap') have bounded clique-width, and*
3. *$\mathcal{M}^\beta$ is bounded.*

[1. and 2. are both satisfied for almost periodic δ]

Simple example

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

Bipartite permutations are defined by $\delta = (2^\infty, \emptyset, 0^\infty)$ where $n^\infty = nnn\cdots$.



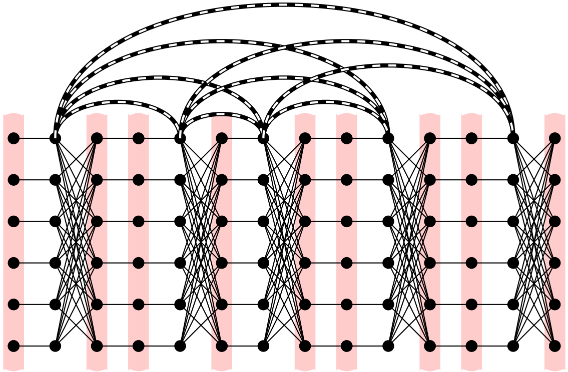
Easy to see:

1. $\mathcal{N}^\delta$ is unbounded and therefore clique-width is unbounded.
2. δ is periodic and therefore recurrent.
3. $\mathcal{M}^\beta$ is bounded and therefore $\mathfrak{G}^\delta$ is a minimal class.

More complex example

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

The Fibonacci word is binary almost periodic and is generated by the substitution $0 \to 01$, $1 \to 0$ (or $S_n = S_{n-1}S_{n-2}$), e.g. $S_5 = 0100101001001$. Let $\delta = (\alpha, \beta, \gamma)$ be a triple such that α is the Fibonacci word, β connects columns i and j when $\alpha_i = \alpha_j = 1$ and γ puts a clique on column i if $\alpha_i = 0$.



Uncountably many

Minimal hereditary classes of graphs of unbounded tree-width and clique-width

The Fibonacci word is an example of a Sturmian word.
Sturmian words are:

- Almost periodic binary.
- Uncountably many that are not *locally isomorphic* (have the same factors).

Lemma (Brignall and C., 2021)

If $\delta_1 = (\alpha_1, \emptyset, 0^\infty)$ and $\delta_2 = (\alpha_2, \emptyset, 0^\infty)$ where α_1 and α_2 are Sturmian words that are not locally isomorphic then the hereditary classes $\mathfrak{G}^{\delta_1}$ and $\mathfrak{G}^{\delta_2}$ are different minimal classes of unbounded clique-width.

$\rightarrow$ uncountably many minimal classes.

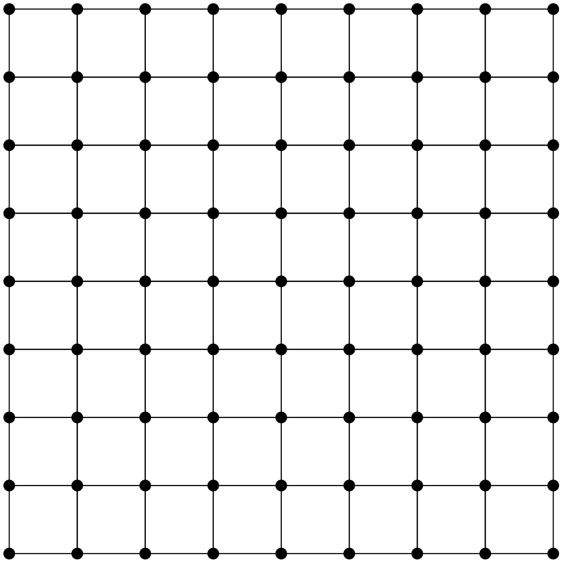
Recap of key points - Clique-width

- In bounded clique-width graph classes many problems which are generally hard become ’easy’ (i.e. an algorithm exists)
- There exist both sparse and dense graphs of bounded clique-width.
- Uncountably many dense minimal classes have been identified.
- We have created a framework that captures all known minimal classes.
- Some necessary characteristics for minimality understood (e.g. recurrence) but they don’t appear to be ’well-behaved’.

5. Tree-width and clique-width in ’sparse’ hereditary classes

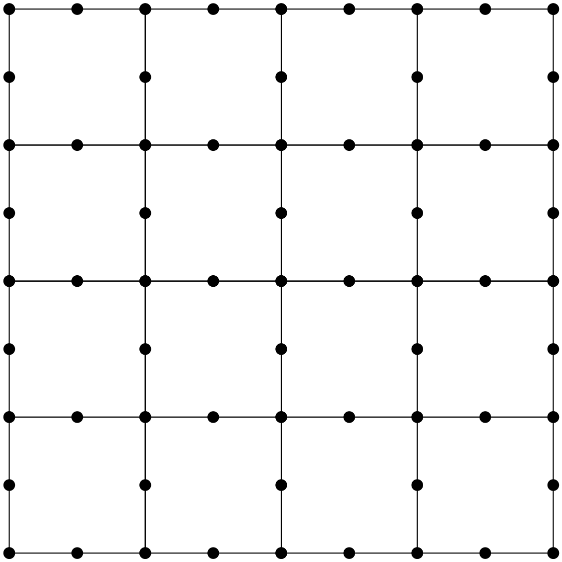
Grids and subdivisions of grids

Minimal hereditary classes of graphs of unbounded tree-width and clique-width



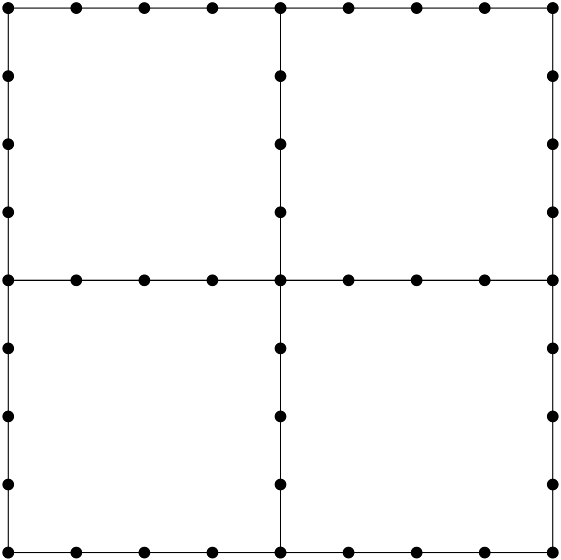
Grids and subdivisions of grids

Minimal hereditary classes of graphs of unbounded tree-width and clique-width



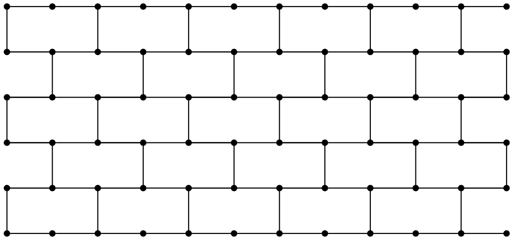
Grids and subdivisions of grids

Minimal hereditary classes of graphs of unbounded tree-width and clique-width



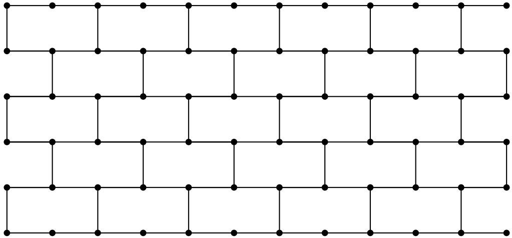
Walls and line graphs of walls

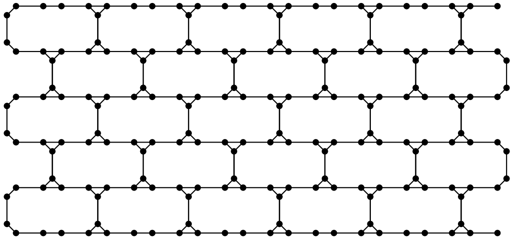
Minimal hereditary classes of graphs of unbounded tree-width and clique-width



Walls and line graphs of walls

Minimal hereditary classes of graphs of unbounded tree-width and clique-width





Thank You For Listening!