Mamadou M. Kanté

UCA - LIMOS - CNRS
31/03/2021

Width Parameters Week

Introduction to Clique-Width
PLAN.

1. CLIQUE- WIDTH DEFINITION
2. SOME GRAPH CLASSES OF (UN)BOUNDED CLIQUE- WIDTH
3. DETOUR TO SOME ALGORITHMIC APPLICATIONS
4. HOW TO HAVE EQUIVALENT MEASURE
5. FROM STRUCTURAL POINT OF VIEW: WHY RANK- WIDTH
**DETOUR TO HR GRAMMAR**

- **R-sourced graph** = graph such that \( \leq R \) vertices are labeled in \( \{k\} = \{1, 2, \ldots, k\} \)

  \[ \text{src} : \{k\} \rightarrow V(G) \] : label injective function

- **Basic Graph**: \( E 

- 3 operations on R-sourced graphs:

<table>
<thead>
<tr>
<th>Forget: ( fgi )</th>
<th>Rename: ( ren_i \rightarrow j )</th>
<th>Fusion: ( | )</th>
</tr>
</thead>
<tbody>
<tr>
<td>No more vertex is labeled ( i ):</td>
<td>No vertex is labeled ( j ):</td>
<td>( G ) and ( H ) R-sourced</td>
</tr>
<tr>
<td>( \text{src}(V(fgi(G))) \subseteq {k} \setminus {i} )</td>
<td>( \text{src}(V(\text{ren}_i \rightarrow j(G))) \subseteq {k} \setminus {i, j} )</td>
<td>( G | H ) : Fuse 1-to-1 sources</td>
</tr>
</tbody>
</table>

\( G \| H = \begin{cases} 1 & \text{if } i = 1 \text{ in } G \text{ or } i = 3 \text{ in } H \text{ and } j = 3 \text{ in } G \text{ or } j = 2 \text{ in } H \text{ and } i, j \text{ are distinct,} \\ 0 & \text{otherwise} \end{cases} \)
Using Basic Graph + 3 Operations one can construct set of terms $T(H_{R_b}, \{\exists y\})$:

- $E$ is a term
- $D_{g_i}(t)$ and $ran_{i-j}(t)$ are terms

- $t_1 \parallel t_2$ is a term
Using Basic Graph + 3 Operations one can construct set of terms $T(\mathcal{HR}_R, \mathcal{F}_A)$:

- $\varepsilon$ is a term: $\text{val}(\varepsilon) = 1$

- $\text{fg}_i(t)$ and $\text{ren}_i \rightarrow_j(t)$ are terms
  
  $\text{val}(\text{fg}_i(t)) = \text{fg}_i(\text{val}(t))$
  
  $\text{val}(\text{ren}_i \rightarrow_j(t)) = \text{ren}_i \rightarrow_j(\text{val}(t))$

- $t_1 \parallel t_2$ is a term
  
  $\text{val}(t_1 \parallel t_2) = \text{val}(t_1) \parallel \text{val}(t_2)$
Theorem A: $\text{Two}(G) \leq \mathbb{R} \iff G \in T(H_{\mathbb{R}^{2+1}}, f_\mathcal{E})$

Proof:

Create edges in bag $\mu$
Theorem A: \( \text{Two}(G) \leq k \iff G \in \mathcal{T}(HR_{2k+1}, e_G) \)

**Proof:**

- Create edges in bag \( \mu \).
- Sources are bags.

 Diagrams showing various graph structures are also present.
Combining Theorem A + Tree Automata + Bodlaender's Theorem

Theorem B (Courcelle '90) Every MSO₂ definable property can be solved in time $O(f{|V|}) \cdot n$, for any $n$-vertex graph $G$.

[decision, search, optimise, count, list versions]

<table>
<thead>
<tr>
<th>MSO₂</th>
<th>MSO₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>* incidences: $\text{inc}(x, e)$</td>
<td>* adjacencies: $E(x, y)$</td>
</tr>
<tr>
<td>* FO + $\exists X$, $X \subseteq E \cup V$</td>
<td>* FO + $\exists X$, $X \subseteq V$</td>
</tr>
</tbody>
</table>
Combining Theorem A + Tree Automata + Bodlaender’s Theorem

Theorem B (Courcelle ’90) Every MSO2 definable property can be solved in time $f(two(a)) \cdot n$, for any $n$-vertex graph $G$.

[decision, search, optimise, count, list versions].

**Natural Question:** Is best in terms of graph classes?
Combining Theorem A +
Tree Automata + Bodlaender's Theorem

Theorem B (Courcelle '90) Every MSO₂ definable property can be solved in time $f(two(n)) \cdot n$, for any $n$-vertex graph $G$.

[decision, search, optimise, count, list versions].

NATURAL QUESTION: IS BEST IN TERMS OF GRAPH CLASSES?

• YES if language MSO₂ is wanted (Courcelle, Seese, ...)
• NO if we restrict language ...
Detour to Co-Graph

- Basic Graph: •
- 2 Operations:
  + Disjoint union: ⊕
  + Complete join: ⊗

\[
\text{val(terms)} = \int \text{co-graphs}
\]
DETOUR TO CO-GRAPHS

- Basic Graph:
- 2 Operations:
  - disjoint union: \( \oplus \)
  - complete join: \( \otimes \)

val (terms) =

co graphs

- A tree-automata can detect when \( E(x, y) \).
- Combining it with Tree Automata techniques:

Theorem C [Folklore]: Every MSO definable property can be decided in linear time, on co-graphs.

Of course hidden HUGE constant
DETOUR TO CO-GRAPHS

Basic Graph:

Operations:
- disjoint union: \( \oplus \)
- complete join: \( \otimes \)

\[ \text{val (terms)} = \text{co graphs} \]

A tree-automata can detect when \( E(x, y) \).
Combining it with Tree Automata techniques:

**Theorem C [Folklore]:** Every MSO definable property can be decided in linear time on cographs.

**BASICS FOR CLIQUE-WIDTH OPERATIONS:**
- Colour vertices
- Add edges between colour classes
1. CLIQUE-WIDTH
- **k-labeled graph** = graph with all vertices labeled with a label in $\Sigma_k$

  $\text{lab} : V(G) \rightarrow \Sigma_k$: labeling function

  $\text{lab}^{-1}(i)$: label class $i$

- **Basic Graph**: 1

- **3 operations**: $\oplus$, $\text{reni} \rightarrow i$, $\text{addi}$

  $\text{addi}(e)$: $i \rightarrow j$

  / all possible edges, no multiple edge

  label$^{-1}(i) \rightarrow$ label$^{-1}(j)$
• k-labeled graph = graph with all vertices labeled with a label in $\mathbb{E}^k$
  \[ \text{lab} : V(G) \rightarrow \mathbb{E}^k \] : labeling function
  \[ \text{lab}^{-1}(i) \] : label class $i$

• Basic Graph : $1$

• 3 operations:
  \[ \oplus, \text{ren}_{i \rightarrow j}, \text{add}_{i,j} : i \neq j \]
  \[ \text{add}_{i,j}(t) : i \neq j \]

\[ \text{VR}_k = \{ \oplus, \text{add}_{i,j}, \text{ren}_{i \rightarrow j} : i, j \text{ in } \{1, 2, \ldots, k\} \} \]

• $\text{val}(t) : k$-labeled graph, $t \in T(\text{VR}_k, \mathbb{E}^k)$
Examples

\( t_2 = \text{add}_{1,2} (1 \oplus 2) \quad \text{val}(t_2) = \quad \)

\( t_n = \text{add}_{1,2} (2 \oplus \text{ren}_{2 \to 1} (t_{n-1})) \quad \text{val}(t_3) = \quad \)
Examples

\[ t_2 = \text{add}_{1,2} (1 \oplus 2) \]
\[ t_n = \text{add}_{1,2} (2 \oplus \text{ren}_{2\rightarrow 1} (t_{n-1})) \]

\[ \text{val}(t_2) : \]
\[ \text{val}(t_3) : \]

\[ \text{val}(t_n) : \]

\[ t' = \text{add}_{1,2} (1 \oplus 1 \oplus 2 \oplus 2) \]
\[ \text{val}(t') : \]

\[ t'' = \text{ren}_{1\rightarrow 3} (t') \]
\[ \text{val}(t'') : \]

\[ t''' = \text{add}_{1,3} (t' \oplus 1) \]
\[ \text{val}(t''') : \]
terms as rooted labeled trees

\[ \rightarrow 1 \]
\[ \rightarrow \text{add}_{i,j}(t) \lor \text{ren}_{i \rightarrow j}(t) \]
\[ \rightarrow t_1 \oplus t_2 \]

Example:
The clique-width of a graph $G$, $cwd(G)$, is the minimum $k$ such that $G \cong \text{val}(t)$, $t \in T(VR_k, \{\#\})$ ($\cong$: standard graph isomorphism).

Do not take into account labels.
The clique-width of a graph \( G, \text{cwd}(G) \), is the minimum \( k \) such that
\[ G \cong \text{val}(t), \quad t \in T(VR_k) \cup F \]
\( (\cong: \text{standard graph isomorphism}) \)

When \( G \cong \text{val}(t) \), we write sometimes \( i(x) \) in \( t \) to refer the vertex \( x \) of \( G \)
it is mapped to by \( x \)
Bound on Clique Width

By definition, \( \text{cud}(G) \leq |V(G)| \). But,

**Theorem 5. (Johansson '98):** \( \text{cud}(G) \leq n - k \) as long as

\[
2^k < n - k
\]

Proof:

1. \( |V_1| = n - k \)
2. \( k \) vertices = \( V_2 \)
3. Use \( n - k \) labels to construct \( G \in V_1 \) and relabel wrt neighbors in \( V_2 \) \( \leq 2^k \) labels
4. Add 1 by 1 vertices in \( V_2 \): one extra label.

\( V_1 \) can be classified in at most \( 2^k \) classes wrt neighborhoods in \( V_2 \).
2. Some Graph Classes of (Un)bounded Clique-Width
Proposition 1: $G$ co graph $\iff G \simeq \text{val}(t)$, $t \in T(VR_a, d^I)$. 

Proof: 

$G_1$ $\overset{\times}{\sim}$ $G_2$ 

\[
\begin{array}{c}
\text{men}_2 \rightarrow 1 \\
\text{add} \rightarrow 1,2 \\
+ \\
t_1 \\
\text{men}_2 \rightarrow 1 \\
t_2
\end{array}
\]
Proposition 1: $G$ co graph $\iff G \cong \text{val}(t), \quad t \in T(VR_2, q1\bar{q})$.

Proof:

The other direction needs a normalisation: when using $\text{add}_{1,2}$, no edges between vertices labeled 1 and those labeled 2.
Paths.

- $\text{cwd}(P_3) = 2$, $\text{cwd}(P_4) = 3$ (not a cograph).
- $\text{cwd}(P_n) = 3$, $n \geq 4$

\[
t_4 = \text{add}_{2,3} \left( \text{ren}_{3 \rightarrow 2} \left( \text{ren}_{2 \rightarrow 1} \left( \text{add}_{1,3} \left( \text{add}_{1,2} (1 \oplus 2) \oplus 3 \right) \right) \oplus 3 \right) \right)
\]

\[
t_{n+1} = \text{add}_{2,3} \left( \text{ren}_{3 \rightarrow 2} \left( \text{ren}_{2 \rightarrow 1} \left( t_n \right) \right) \oplus 3 \right)
\]

$P_{n-1}$
Cycles

- $\text{cwdl}(C_4) = 2$ (co-graph)
- $\text{cwdl}(C_5) = 3$
- $\text{cwdl}(C_n) \leq 4$
Distance-HEREDITARY GRAPHS

G is DH $\iff$ G can be obtained from a single vertex by adding twins on pendant vertices.

Proposition 2.1 [Golumbic, Rotics'00]: $cwd(DH) \leq 3$.

Inductive Construction: Let's use the color 1, 2, 1:

1. is never used to create an edge.
2. $t_i$ generates $G[x_1, \ldots, x_i]$; $x_{i+1}$ is twin/pendant of $x_j$.
3. $t_{i+1} = t_i[i(x_j)/t']$ where $t'$ is:

$$t' = \frac{x_j}{\text{false twins}}$$
Distance-HEREDITARY GRAPHS

$G$ is DH $\iff$ $G$ can be obtained from a single vertex by adding twins or pendant vertices.

**Proposition 2.1** [Golumbic, Rotics '00]: \text{cwd}(DH) \leq 3.

Inductive Construction: Let's use the color $1,2, \top$:

1. $\top$ is never used to create an edge.
   - $t_i$ generates $G[x_1, \ldots, x_i]$, $x_{i+1}$ is twin/pendant of $x_j$
   - $t_{i+1} = t_i[i(x_j)/t]$ where $t'$ is

$t$:

```
  i(x_0)    i(x_{i+1})
    \downarrow
   \top
```

false twins

```
  i(x_0)    i(x_{i+1})
    \downarrow
  \text{add } i \rightarrow
```

true twins
Distance-HEREDITARY GRAPHS

Distance-HEREDITARY GRAPHS

**Proposition 2.1 (Golumbic, Rotics '00):** $\text{cwd}(\text{DH}) \leq 3$.

**Inductive Construction:** Let's use the color 1, 2, 1:

- 1 is never used to create an edge.
- $t_i$ generates $G[x_1, \ldots, x_i]$, $x_{i+1}$ is twin/pendant of $x_j$.
- $t_{i+1} = t_i[t'/i[x_j]]$ where $t'$ is

\begin{itemize}
  \item false twins
    \[
    t \xrightarrow{\text{add}_1 i}
    \]
    \[
    i(x_0) \xrightarrow{\text{add}_i i} i(x_i)
    \]
    \[
    i(x_0) \xrightarrow{\text{add}_i i} \text{true twins}
    \]
  \item true twins
    \[
    t \xrightarrow{\text{add}_1 i}
    \]
    \[
    i(x_0) \xrightarrow{\text{add}_i i} i(x_i)
    \]
    \[
    i(x_0) \xrightarrow{\text{add}_i i} \text{pendant}
    \]
\end{itemize}
Proposition 2.2: \( \text{cwd}(G) \leq 2^{\text{tw}(G)+1} \)

Proof: \((T, \psi)\) a tree-decomposition of width \(\text{tw}\). Do a proper \((\text{tw}+1)\)-coloring of \(G\).
Proposition 2.2: \( \text{cwd}(G) \leq 2^{tw(G)+1} \)

Proof: Let \((T, f)\) a tree-decomposition of width \( tw \).

1. Do a proper \((tw+1)\)-coloring of \( G \).
2. If \( u \in V(T) \), every edge between \( V_u = \bigcup_{v \leq u} f(v) \) and \( V(G) \setminus V_u \subseteq G[f(u)] \).
3. For \( u \in T \), compute \( t_u \) for \( G[V_u \setminus f(\text{parent}(u))] \) such that each vertex has label \( \text{color}(y) : y \in f(u) \setminus f(\text{parent}(u)) \) for \( xy \in E(G) \).
Proposition 2.2: \( \text{cwd}(G) \leq 2^{\text{tw}(G)} + 1 \)

Proof: Let \( f \) be a tree-decomposition of width \( tw \).

1. Do a proper \((tw+1)\)-coloring of \( G \).
2. If \( u \in V(f) \), every edge between \( V_u = \bigcup_{v \in u} f(v) \) and \( V(G) \setminus V_u \subseteq G[f(u)] \).
3. For each \( u \), compute \( t_u \) for \( G[V_u \setminus f(\text{parent}(u))] \) such that each vertex has label \( \phi \).
4. \( \text{color} \subset y \subset \phi \subset f(\text{parent}(u)), xy \in E(G) \) \}

\[
\begin{align*}
\tau_a & = \text{ren}_{2 \rightarrow 1} (2) \\
\tau_d & = \text{ren}_{1 \rightarrow 3,4,f} (\text{ren}_{2 \rightarrow \phi} (\text{add}_{1,2} (1 + \tau_a))) \\
\tau_b & = \text{ren}_{3 \rightarrow 7,1,2} (3) \\
\tau_c & = \text{ren}_{2 \rightarrow 3,4} (\text{ren}_{1 \rightarrow 3,4} (\text{add}_{1,2,3,1} (\text{add}_{1,2,3,1} (\text{add}_{1,2} (\text{add}_{1,2} (\tau_d + 1 + 2)))))))
\end{align*}
\]
**Proposition 2.2**: \( \text{cowd}(G) \leq \text{tw}(G) + 1 \)

**Proof**:  
1. \((T, \pi)\) a tree-decomposition of width \(\text{tw} \)
2. Do a proper \((\text{tw} + 1)\)-coloring of \(G\)
3. If \(u \in V(T)\), every edge between \(V_u = \bigcup_{v \leq u} \pi(v)\) and \(V(G) \setminus V_u \subseteq \pi(u)\).
4. For each \(u\), compute \(t_u\) for \(G \in V_u \mid \pi(\text{parent}(u))\)
such that each vertex has label.
   
\[
\text{color}(y) = y \in \pi(u) \cap \pi(\text{parent}(u)), xy \in E(G) \}
\]

\[
\begin{align*}
t_a &= \text{ren}_{2 \rightarrow 1}(2) \\
t_b &= \text{ren}_{1 \rightarrow 3,4,5}(\text{ren}_{2 \rightarrow 1}(\text{add}_{1,2}(1 + \text{ta}))) \\
t_c &= \text{ren}_{2 \rightarrow 3,4,5}(\text{add}_{1,3,4,5}(1 + \text{ta})))
\end{align*}
\]

\[
\begin{align*}
t_\ell &= \text{add}_{3,4,5,6}(\text{add}_{3,4,5,6}(t_\ell \oplus t_c \oplus t)) \\
t &= \text{add}(i,j)
\end{align*}
\]
Proposition 2.2: \( \text{cwd}(G) \leq \frac{\text{tw}(G) + 1}{2} \)  
\( \forall K \in G, \text{tw}(G) = k, \text{cwd}(G) \geq \lceil \frac{k}{2} \rceil - 1 \)

Proposition 2.3: \( \text{cwd}(\text{line}(G)) \leq P(\text{tw}(G)) \)

Proposition 2.4: On \( K_{p,p} \)-subgraph free graphs,  
\( f(\text{tw}(G)) \leq \text{cwd}(G) \leq g(\text{tw}(G)) \)
Some Operations on Graphs

- $H \subseteq G$: induced subgraph $c wd(H) \leq c wd(G)$
Some Operations on Graphs

- $H \subseteq G$: induced sub graph
  $\text{cwd}(H) \leq \text{cwd}(G)$

- $\overline{G} = \text{complement of } G$
  $\text{cwd}(\overline{G}) \leq 2 \cdot \text{cwd}(G)$

- any term in $T(V^R, \{r, s\})$ is equivalent to
  some using binary $\bigoplus_R^f : R \subset [k] \times [k], f : [k] \rightarrow [k]$ edges added between operands by $R$

- easy now to construct $\overline{G}$: replace $\bigoplus_R^f$ by $\bigoplus_{\overline{R}}^f$
- any term with $\bigoplus_R^f$ translates into one in $T(V^R, \{r, s\})$
Some Operations on Graphs

- \( H \subseteq G \): induced subgraph
  \[ \text{cwd}(H) \leq \text{cwd}(G) \]

- \( \overline{G} \): complement of \( G \)
  \[ \text{cwd}(\overline{G}) \leq 2 \times \text{cwd}(G) \]

- Substitution: \( v \in V(G) \), \( V(G) \cap V(H) = \emptyset \)
  \( G \in G[H] \)
  \[ V(G^-) = V(G) \cup V(H) \setminus \{v\} \]
  \[ E(G^-) = E(G \setminus v) \cup E(H) \cup \Delta_G(v) \times V(H) \]

\( V(H) \) is a module in \( G^- \)
Some Operations on Graphs

- \( H \subseteq G \): induced subgraph
  \[ \text{cwd}(H) \leq \text{cwd}(G) \]
- \( \overline{G} = \text{complement of } G \)
  \[ \text{cwd}(\overline{G}) \leq 2 \cdot \text{cwd}(G) \]

Substitution: \( v \in V(G), V(G) \cap V(H) = \emptyset \)
\[ G \cong G[H/v] \]

\[ V(G') = V(G) \cup V(H) \]
\[ E(G') = E(G) \cup E(H) \cup N_G(v) \times V(H) \]

\( V(H) \) is a module in \( G' \)

Observation 2.1: \( \text{cwd}(G') = \max \{ \text{cwd}(G), \text{cwd}(H) \} \)
\[ \text{val}(t_G) \cong G, \quad \text{val}(t_H) \cong H \]
\[ \text{val} \left( t_G \left[ \left( \bigcup_{n \in \mathbb{N}} \left( t_H \right) \right) / l(v) \right] \right) \cong G' \]
Some Operations on Graphs

- $H \subseteq G$: induced subgraph
  \[ \text{cwd}(H) \leq \text{cwd}(G) \]
- $\overline{G}$ = complement of $G$
  \[ \text{cwd}(\overline{G}) \leq 2 \cdot \text{cwd}(G) \]

Substitution:
- $V(G), V(G) \cap V(H) = \emptyset \implies G \cong G[H/\sigma]$
- $V(G') = V(G) \cup V(H) \setminus \sigma$
- $E(G') = E(G[\sigma]) \cup E(H) \cup \sigma E(H) \cup \sigma V(H)$

$V(H)$ is a module in $G'$

Observation 2.1
- $\text{cwd}(G') = \max \left\{ \text{cwd}(G), \text{cwd}(H) \right\}$
- $\text{val}(t_G) \cong G$, $\text{val}(t_H) \cong H$
- $\text{val}(t_G \left[ (0, (t_H)) \right] / d(v)) \cong G'$

Theorem 2.1: For every graph $G$
- $\text{cwd}(G) = \max \left\{ \text{cwd}(H) : H \subseteq G \text{ prime} \right\}$
  every module is either all or a singleton.
Corollary of Theorem 2.1

Theorem 2.1: For every graph $G$

$$\text{cwd}(G) = \max \{ \text{cwd}(H) : H \subseteq G \text{ prime } \}$$

Any hereditary graph class with prime graphs of bounded clique width has bounded clique-width.

Examples: $(P_5, \Delta)$-free, chordal $(\cdot \cdots \cdots)$-free, $(\circ, \bullet)$-free, $(\circ, \circ)$-free, $(P_5, \bigcirc)$-free, ...

Complete classification for those excluding a one-vertex extension of $P_4$, graphs of size $\leq 4$

non perfect $(4K_1, C_4, C_5)$-free, ...
Some graph classes have undecidable MS$_1$-theory. Yet, others can encode some with undecidable theories. If $\text{Cwd}(G) = k \iff G \approx \text{val}(t)$, $t \in T(\forall R_k, \exists t)$

- $t$ a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

By tree-automata $\equiv$ MSOL on trees, and techniques to compute tree-automata from MSOL formulas, we have:
**UNBOUNDED CLIQUE-WIDTH**

A) **SUPER-LOGIC**

Some graph classes have undecidable $\Pi_1$-$\text{MSO}_1$-theory. Yet, others can encode some with undecidable theories.

- If $\text{cwd}(t) = k \Leftrightarrow G \cong \text{val}(t), \ t \in T(V_{R_{k}}, P_{t})$

  - $t$ a rooted labeled tree
  - A bottom-up tree-automata to decide whether $xy \in E(G)$

  $\Rightarrow$ By tree-automata $\equiv \Pi_{1}$-$\text{MSO}_1$ on labeled trees, and techniques to compute tree-automata from $\Pi_{1}$-$\text{MSO}_1$ formulas, we have:

For every $\Pi_{1}$-$\text{MSO}_1$ formula $\varphi$, there is a deterministic tree-automata $A_{\varphi}$ such that

$$G \models \varphi \iff t \in L(A_{\varphi})$$
Some graph classes have undecidable $\forall$S. -Theory.
Yet, others can encode some with undecidable theories.

If $\text{cwd}(G) = k \iff G \equiv \forall \text{val}(t), t \in T(VR_k, \\psi)$

- If a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

By tree-automata $\equiv$ MSOL on labeled trees, and techniques to compute tree-automata from MSOL formulas, we have:

For every MSOL formula $\Phi$, there is a deterministic tree-automata $A_\Phi$ such that $G \models \Phi \iff t \in L(A_\Phi)$

MSOL decidable on graphs of clique-width $k$. 
**UNBOUNDED CLIQUE-WIDTH**

A) **SUPER-LOGIC**

Some Graph classes have undecidable MSO$_1$-Theory. Yet, others can encode some with undecidable theories.

- If $\text{Cwd}(G) = k \iff G \cong \text{valit}(t), t \in T(VR_k, \Pi)$
- If a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

By tree-automata $\equiv$ MSO$_1$ on labeled trees, and techniques to compute tree-automata from MSO$_1$ formulas, we have:

For every MSO$_1$ formula $\varphi$, there is a deterministic tree-automata $T_A$ such that

$G \models \varphi \iff t \in L(T_A)$

**Examples:**
- All graphs
- Grids
- Planar graphs of degree $\leq 3$
- Split graphs
- Bipartite graphs

MSO$_1$ decidable on graphs of clique-width $\leq k$. 

\[ \text{Cwd}(G) = k \iff G \cong \text{valit}(t), t \in T(VR_k, \Pi) \]
A SUPER-LOGIC

Some graph classes have undecidable MSO-theory. Yet, some can encode some with undecidable theories.

- If \( \text{Cwd}(G) = k \iff G \cong \text{val}(t), t \in T(V R_k, \Pi) \)
- A rooted and labeled tree
- A bottom-up tree-automata to decide whether \( xy \in E(G) \)

By tree-automata \( = \text{MSOL on labeled trees} \), and techniques to compute tree-automata from MSOL formulas, we have:

For every MSOL formula \( \varphi \), there is a deterministic tree-automata \( A \) such that

\[ G \models \varphi \iff t \in L(A \varphi) \]

MSO, decidable on graphs of clique-width \( k \).

Examples:
- All graphs
- Grids
- Planar graphs of degree \( \leq 3 \)
- Split graphs
- Bipartite graphs encoding

But bounds are bad
(B) Use Hands and Case analysis

B.1: Square Grids

\[ \text{cwl}(G_{n,n}) = n+1 \]

Upper Bound

Lower bound

Do case analysis depending on whether a subtree contains a full row or column.
UNBOUNDED CLIQUE-WIDTH

(B) Use Hands and Case analysis

B.2 Unit interval graphs

\[ \text{universal unit-interval graph} \]

\[ cwd = n + 1 \]
Use Hands and Case analysis

B.3. Split graphs

For any term, there is node \( \mu \) such that
\[
\frac{n(n+1)}{6} \leq |V(G_n)| \leq \frac{n(n+1)}{3}
\]
Look at the edges in the other side incident with it: counting will contradict that \( \text{cwd} < n/72 \).
Some difficulties to deal with clique-width

 redistribution

\[ C_{\geq} : \text{the only monotone operation.} \]

- The audience knows how hard it can be to study structure with \( C_{\geq} \).
- Few graph classes are \( wqo \) under \( C_{\geq} \).
- Computing such parameters with broad algorithmic applications is \( NP \)-Hard.
- One can expect \( FPT \) polynomial time.
- But, particularly hard for clique-width.
- Only one polynomial time algorithm: \( \text{cwd}(G) \leq 3 \).
- One can compute exactly for Bounded treewidth.
3. Detour to Some Algorithmic Applications

- Courcelle et al. Theorem give intractable
  FPT algorithm:
- Your own DP if you want a better one.

Assume a term is given
3.1. Independent Set

Do a bottom-up traversal and compute for $\mu$ and $\lambda \subseteq \mathbb{R}$

$$\text{tab}_{\mu}[\lambda] = \max \left\{ X : \text{tab}^\mu_0(X) = X \right\}$$

- $\mu_1$:
  $$\text{tab}_{\mu}[\lambda] = \max \left\{ \text{tab}_{\mu_1}[\lambda], \text{tab}_{\mu_2}[\lambda] \right\}$$

- $\mu_2$:
  $$\text{tab}_{\mu}[\lambda] = \left\{ \begin{array}{ll}
  \text{tab}_{\mu_1}[\lambda] & \text{if } j \notin \lambda \\
  \max \left\{ \text{tab}_{\mu_1}[\lambda], \text{tab}_{\mu_2}[\lambda] \right\} & \text{otherwise}
  \end{array} \right.$$
3.1. Independent Set

- Do a bottom-up traversal and compute for $\mu$ and $\lambda \in [k]$

\[ \text{tab}_\mu[\lambda] = \max \left\{ \right. \left. X : \text{tab}_\mu(X) = X \right\} \]

- $\mu:\text{tabula}=\max\{\text{tab}_\mu, \text{tab}_\mu[U] \}$

- $i \in \lambda : \text{tab}_\mu[\lambda] = \max \left\{ \text{tab}_\mu[U], \text{tab}_\mu[\lambda \setminus \{i\}] \right\}$

- $i \in \lambda : \text{tab}_\mu[\lambda] \left\{ \right. \left. \right\} \text{tab}_\mu[U] \text{if } j \notin \lambda \}

- $\text{tab}_\mu[\lambda] = \emptyset \text{ if } \{i, j\} \subseteq \lambda$

- time: $= 2^{k+1} n \text{ if } t \text{ given}$

- $2^k$: optimal under ETH
As for the previous algorithm, keep the labels for each colouring.

Assume you want to check whether $G$ is $2$-colorable.

For each node $u$, keep the following for $G_u$:

For each proper $2$-coloring $(X_1, X_2, \ldots, X_d)$

$$(C_1, C_2, \ldots, C_d) : C_i = \{j \in E : i \in \text{lab}(X_j)\}$$

At most $2^{|E|}$ possible entries

We update similarly as previous algorithm.
3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each colouring.

- Assume you want to check whether $G$ 2-colorable.

- For each node $u$, keep the following for $G_u$:
  - for each proper 2-coloring $(X_1, X_2, ..., X_d)$
  - $(G_1, G_2, ..., G_k): G_i = \{ j \in \mathbb{Z}_d : i \in \text{lab}(X_j) \}$

- At most $2^d \cdot k$ possible entries

- We update similarly as previous algorithm

- A clever one with $(2^d - 2)^k$ possible entries.
  No $(2^d - 2 - \varepsilon)^k$ one under SETH $\forall \varepsilon > 0$
3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each colouring.
- For a proper coloring $\chi = (X_1, X_2, \ldots, X_\ell)$, associate the following function:
  \[ \ell \mapsto \{ x_j : \text{lab}(x_j) = \ell \} \]
3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each colouring.

- For a proper coloring $\mathbf{X} = (X_1, X_2, \ldots, X_k)$, associate the following function:
  \[
  L \mapsto [\mathbf{X}] \quad \text{where} \quad \forall j \in \left[ \text{lab}(X_j) = L \right]
  \]

- $k \leq n \Rightarrow n^k$ possible such functions.

- The update is the same as previously.
3.8. Chromatic Number

• As for the previous algorithm, keep the labels for each coloring.

• For a proper coloring $\chi = (X_1, X_2, \ldots, X_d)$, associate the following function:

$$\chi : L \rightarrow [2^k]$$

$$L_j : \text{lab}(x_j) = L_j$$

• $d \leq n \Rightarrow 2^k$ possible such functions.

• The update is the same as previously.

• Optimal under ETH: no $n^{o(\log k)}$
3.3 Hamiltonian Path

• If \( P \) is a Hamiltonian path, and \( u \) a node, \( PN \{G_u\} \) is a collection of paths.

• Construct the multigraph \( M(P) \) on \( \{e_k\} \)

\[
\text{# of subpaths from } \quad i \xrightarrow{\text{lab}(i)} j \quad \text{to} \quad j \xrightarrow{\text{lab}^{-1}(j)} i
\]

• \( n^2 \) possible such multigraphs.

• \( P \cup Q \) a Hamiltonian Path \( \iff \) an alternating Eulerian tour on \( M(P) \cup M(Q) \).
3.3 HAMILTONIAN PATH

• If P is a HAM PATH, and u a node,
  $P \cap V(G_u)$ is a collection of paths.

• Construct the multigraph $M(P)$ on $E_k$

  \[
  \text{# of subpaths from } \quad \frac{\text{i \ in } \text{lab}(i) \text{ to } \text{lab}^{-1}(j)}{\text{\ for all } j}
  \]

• $N^k$ possible such multigraphs.

• PU a HAM PATH (\Rightarrow) an alternating eulerian
  tour on $M(P) \cup M(Q)$.

• $P_1 \equiv P_2$ if $M(P_1)$ and $M(P_2)$
  • Same connected components
  • Same degree sequence
### 3.3 Hamiltonian Path

- If $P$ is a Hamiltonian path, and $u$ a node, $P \cup V(G_u)$ is a collection of paths.

- Construct the multi graph $\mathcal{M}(P)$ on $[k]$.

  \[
  \text{# of subpaths from } \quad i \rightarrow \mathcal{b}(i) \text{ to } \mathcal{b}^{-1}(j). \quad j
  \]

- $N^k$ possible such multi-graphs.

- $P \cup Q$ a Hamiltonian path ($\Rightarrow$) an alternating Eulerian tour on $\mathcal{M}(P) \cup \mathcal{M}(Q)$.

- $P_1 \approx P_2$ if $\mathcal{M}(P_1)$ and $\mathcal{M}(P_2)$
  - Same connected components
  - Same degree sequence

- $N^k$ possible entries, no $n^{o(k)}$ under ETH.
Many other examples

- Any domination-like problem with degrees on finite/co-finite subsets of \( \mathbb{N} \)
- Computation of graph polynomials.

...
4. Some equivalent measures
Look at Neighborhoods

important, how the neighborhoods are

Layout
Look at Neighborhoods

important: how the neighborhoods are any function describing it is good

Layout
- CWD: number of twin classes (with a symmetry).
Look at Neighborhoods

Important: how the neighborhoods are any function describing it is good

Layout
- CUD: number of twin classes (with a symmetry).
- Others: distinct neighborhoods, rank-width, boolean-width, H-join, etc.
5. Why RANK-WIDTH
\[ \forall e \in E(T) : \begin{array}{c|ccc} x & y & z \\ \hline u & 0 & 0 & 0 \\ v & 0 & 0 & 0 \\ v & 1 & 0 & 0 \end{array} \]

\[ w(e) = \text{rk} (A[X_e, X_e]) \]

**RANK-WIDTH**

**Layout**: \((T, L), L : V(G) \rightarrow \text{leaves of } T \)
RANK-WIDTH

\[ \forall e \in E(T) : \begin{array}{ccc}
  x & y & z \\
  u & 0 & 0 \\
  v & 0 & 0 \\
  w & 1 & 0 \\
\end{array} \]

\[ A[X_e, X_e] \]

\[ w(e) = \text{rk} ( A[X_e, X_e] ) \]

\[ \text{wd}(T, L) = \max_{e \in E(T)} \frac{1}{2} w(e) \]

Layout : \((T, L), L : V(T) \to \text{leaves of } T\)
**Rank-Width**

For all $e \in E(T)$:

$\forall e \in E(T): \begin{array}{c|ccc}
\chi & \gamma & \omega & z \\
\hline
u & 0 & 0 & 0 \\
v & 0 & 0 & 0 \\
w & 1 & 0 & 0 \\
\end{array}$

$A[e, \overline{e}] = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
\end{bmatrix}$

$w(e) = \text{rk}(A[e, \overline{e}])$

$\text{wd}(T, L) = \max_{e \in E(T)} \frac{1}{2} w(e)$

$\text{wd}(G) = \min_{(T, L) \text{ layout}} \text{wd}(T, L)$

**Layout**: $(T, L)$, $L: V(G) \rightarrow \text{leaves of } T$
Proposition 5.1 \[ \text{rwd}(G) \leq \text{cwd}(G) \leq 2^{\text{rwd}(G)+1} \]

Proof: \((T,L )\) a layout, root it at \( r = \text{subdivide an edge} \)

For \( u \in V(T) \), compute \( tu \) for \( G_u \) s.t.

\[ \text{lab}(x) = \text{lab}(y) \iff N(x) \cap V(G_u) = N(y) \cap V(G_u) \]
Proposition 5.1 \[ \text{rank-width} \]

For \( \gamma \in \mathbb{N} \), let \( \gamma \) be a layout, root it at \( r \).

\[ \forall \mu \in V(T), \text{ compute } t_{\mu} \text{ for } G \mu \text{ s.t. } \]

\[ \text{deg}(x) = \text{deg}(y) \iff N(x) \cap V(G \mu) = N(y) \cap V(G \mu) \]

at most \( 2^\text{rank-width} \) such twin classes.

\[ t_{\mu} = \bigcirc \text{ add } (t_{\mu} \oplus \text{ Oren } i \oplus (t_{\mu})) \]

\[ (i, j') \in R_{\mu} \]

\[ R_{\mu} = \{(i, j') : \exists x \in G \mu, y \in G \mu, xy \in E, \text{ labeled resp. } i \text{ and } j \} \]
Proposition 5.1 \( rwd(G) \leq \omega d(G) \leq 2^{rwd(G) + 1} - 1 \)

Proposition 5.2 \( rwd(DH) = 1 \)

Proof: same proof idea as for clique-width. \( \square \)
Proposition 5.1
\[ \mathrm{rwd}(G) \leq \omega_d(G) \leq 2^{\mathrm{rwd}(G)+1} - 1 \]

Proposition 5.2
\[ \mathrm{rwd}(DH) = 1 \]

Proposition 5.3
\[ \mathrm{rwd}(G) = \max_{H \leq G} \{ \mathrm{rwd}(H) \} \] for a prime with split decomposition.
Proposition 5.1 \[ \text{rw}d(G) \leq \text{cwd}(G) \leq 2 \]

Proposition 5.2 \[ \text{rw}d(DH) = 1 \]

Proposition 5.3 \[ \text{rw}d(G) = \max_{H \subseteq G} \text{rw}d(H) \] (prime w.t. split-decomposition)

Observation 5.1 \[ H \subseteq G, \text{ then } \text{rw}d(H) \leq \text{rw}d(G) \]

Proposition 5.4 \[ \text{rw}d(G) \leq \text{tw}(G) + 1 \]
Proposition 5.1: \[ \text{rwd}(G) \leq \text{cwd}(G) \leq 2^{\text{rwd}(G) + 1} - 1 \]

Proposition 5.2: \[ \text{rwd}(DH) = 1 \]

Proposition 5.3: \[ \text{rwd}(G) = \max \{ \text{rwd}(H) \mid H \leq_i G \} \]

Observation 5.4: If \( H \leq_i G \), then \( \text{rwd}(H) \leq \text{rwd}(G) \).

But more operations.
- Adding a row to other rows does not increase rank.
- Similarly for column.

\( \begin{bmatrix} x & y \\ \end{bmatrix} \)
RANK WIDTH and STRUCTURE

- Adding a row to other rows does not increase rank.
  Similarly for column.

\[
\begin{bmatrix}
\end{bmatrix}
\]

\[
\begin{array}{cccccccc}
\text{x} & \text{y} & \text{z} & \text{w} & \text{v} & \text{u} \\
\hline
\text{x} & 0 & 1 & 1 & 1 & 0 & 0 \\
\text{y} & 1 & 0 & 1 & 0 & 0 & 0 \\
\text{z} & 1 & 1 & 0 & 0 & 0 & 0 \\
\text{w} & 1 & 0 & 0 & 0 & 1 & 0 \\
\text{v} & 0 & 0 & 0 & 1 & 0 & 0 \\
\text{u} & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]
**RANK-WIDTH and STRUCTURE**

- Adding a row to other rows does not increase rank.
- Similarly for column.

\[
\begin{bmatrix}
? & ? \\
? & ?
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
<th>v</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x+y</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x+z</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x+w</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x+m</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[G \ast x = \text{local complementation at } x\]
RANK-WIDTH and STRUCTURE

- adding a row to other rows does not increase rank.
- similarly for column.

\[ \begin{bmatrix} x \end{bmatrix} \]

\[
\begin{array}{cccccccc}
\hline
x & y & z & w & u & v & m \\
\hline
x & 0 & 1 & 1 & 1 & 0 & 0 \\
x+y & 1 & 0 & 0 & 1 & 0 & 0 \\
x+z & 1 & 0 & 0 & 1 & 0 & 0 \\
x+w & 1 & 1 & 1 & 0 & 1 & 0 \\
x+m & 0 & 0 & 0 & 1 & 0 & 0 \\
\hline
\end{array}
\]

\[ G \ast x = \text{local complementation at } x \]

\[ G \ast x = (G \setminus N(x)) \cup G \upharpoonright N(x) \]

\[ Y \]

\[ Z \]

\[ x \]

\[ w \]

\[ m \]
RANK-WIDTH and STRUCTURE

- adding a row to other rows does not increase rank
- similarly for column.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
<th>v</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x+y</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x+z</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x+w</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x+u</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

local complementation at x

\[ G \# x = (G \setminus N(x)) \cup (G[N(x)]) \]

H vertex-minor of G if H \subseteq G',

G' = successive local complementations on G.
Proposition 5.5: If \( H \) is a vertex-minor of \( G \), then \( \text{pwd}(H) \leq \text{pwd}(G) \).
Proposition 5.5: If \( H \) is a vertex-minor of \( G \), then \( \text{pwd}(H) \leq \text{pwd}(G) \).

Proof sketch.

Old notion: already used by Bouquet to characterize circle graphs by a finite list of obstructions.
Graphs of bounded rank-width are wqo by vertex-minor and have no infinite anti-chain.
Graphs of bounded rank-width are wqo by vertex-minor:
no infinite anti-chain.

Every hereditary class of bounded rank-width is characterized by a finite list of obstructions.
Graphs of bounded rank-width are wqo by vertex-minor.
no infinite anti-chain.

Every hereditary class of bounded rank-width is characterized by a finite list of obstructions.

One has a bound on the size of obstructions for rank-width k.
Graphs of bounded rank-width are wqo by vertex-minor, no infinite anti-chain.

Every hereditary class of bounded rank-width is characterized by a finite list of obstructions.

One has a bound on the size of obstructions for rank-width $k$:

- Vertex-minor is $CMS_1$-definable

$\Rightarrow$ One can recognize graphs of rank-width $k$. 

Graphs of bounded rank-width are wqo by vertex-minor
no infinite anti-chain.

Every hereditary class of bounded rank-width
is characterized by a finite list of obstructions.

One has a bound on the size of obstructions
for rank-width $k$:
- vertex-minor is $CMS_1$-definable
  $\Rightarrow$ one can recognize graphs of rank-width $k$.

A combinatorial recognition algorithm
Graphs of bounded rank-width are wqo by vertex-minor.

Every hereditary class of bounded rank-width is characterized by a finite list of obstructions.

One has a bound on the size of obstructions for rank-width $k$:
- Vertex-minor is $C_{175}$-definable
- One can recognize graphs of rank-width $k$.

A combinatorial recognition algorithm

The rank function is submodular and symmetric

$\Rightarrow$ Obstruction set system like brambles = tangles
Graphs of bounded rank-width are wqo by vertex-minor.

• Every hereditary class of bounded rank-width is characterized by a finite list of obstructions.
• One has a bound on the size of obstructions for rank-width $k$:
  - vertex-minor is $\text{CMS}_4$-definable
  $\Rightarrow$ One can recognize graphs of rank-width $k$.
• A combinatorial recognition algorithm
• The rank function is submodular and symmetric
  $\Rightarrow$ Obstruction set system like brambles.
• HUN: Looks like a lot tree-width, but suited for dense graphs.
Graphs of bounded rank-width are wqo by vertex-minor
no infinite anti-chain.

Every hereditary class of bounded rank-width
is characterized by a finite list of obstructions.

One has a bound on the size of obstructions
for rank-width $k$:
- vertex-minor is $C^m_{1,1}$-definable
  $\Rightarrow$ One can recognize graphs of rank-width $k$.

A combinatorial recognition algorithm

The rank function is submodular and symmetric
$\Rightarrow$ Obstruction set system like brambles.

HUN: Looks like a lot tree-width, but suited
for dense graphs.

YES INDEED GENERALISE PRELIMINARY RESULTS FROM
GRAPH/MATROID MINOR THEORY
MERCI