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

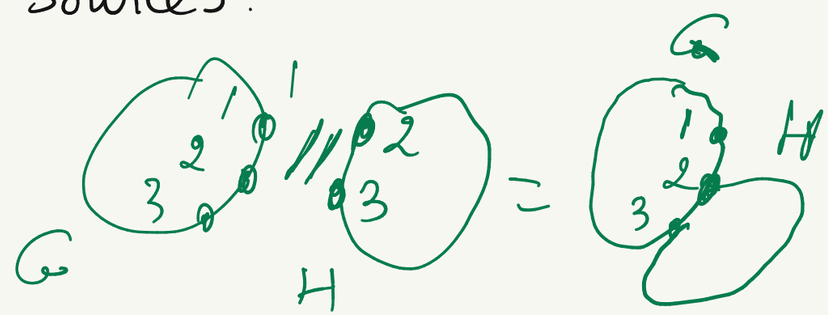
DETOUR TO HR GRAMMAR

- k-Sourced graph = graph such that $\leq k$ vertices are labeled in $[k] = \{1, 2, \dots, k\}$

  $src : [k] \longrightarrow V(G)$: label injective function

- Basic Graph : $\exists$ 

- 3 operations on k-Sourced graphs :

Forget : f_{g_i}	Rename : $ren_{i \to j}$	Fusion : $//$
no more vertex is labeled i : $SRC^{-1}(V(f_{g_i}(G))) \subseteq [k] \setminus \{i\}$	• No vertex is labeled j. • Label $SRC(i)$ into j. $SRC^{-1}(V(ren_{i \to j}(G))) \subseteq [k] \cup \{j\} \setminus \{i\}$	• G and H k-sourced • $G // H$: FUSE 1-to-1 sources. 

Using Basic Graph + 3 Operations one can construct set of terms $T(HR_R, \{E_i\})$:

- E is a term
- $\Box_i(t)$ and $ren_{i \to j}(t)$ are terms
- $t_1 // t_2$ is a term

Using Basic Graph + 3 Operations one can construct set of terms $T(HR_R, \{E_i\})$:

- E is a term : $val(E) =$ 

- $fg_i(t)$ and $ren_{i\to j}(t)$ are terms

$val(fg_i(t)) = fg_i(val(t))$

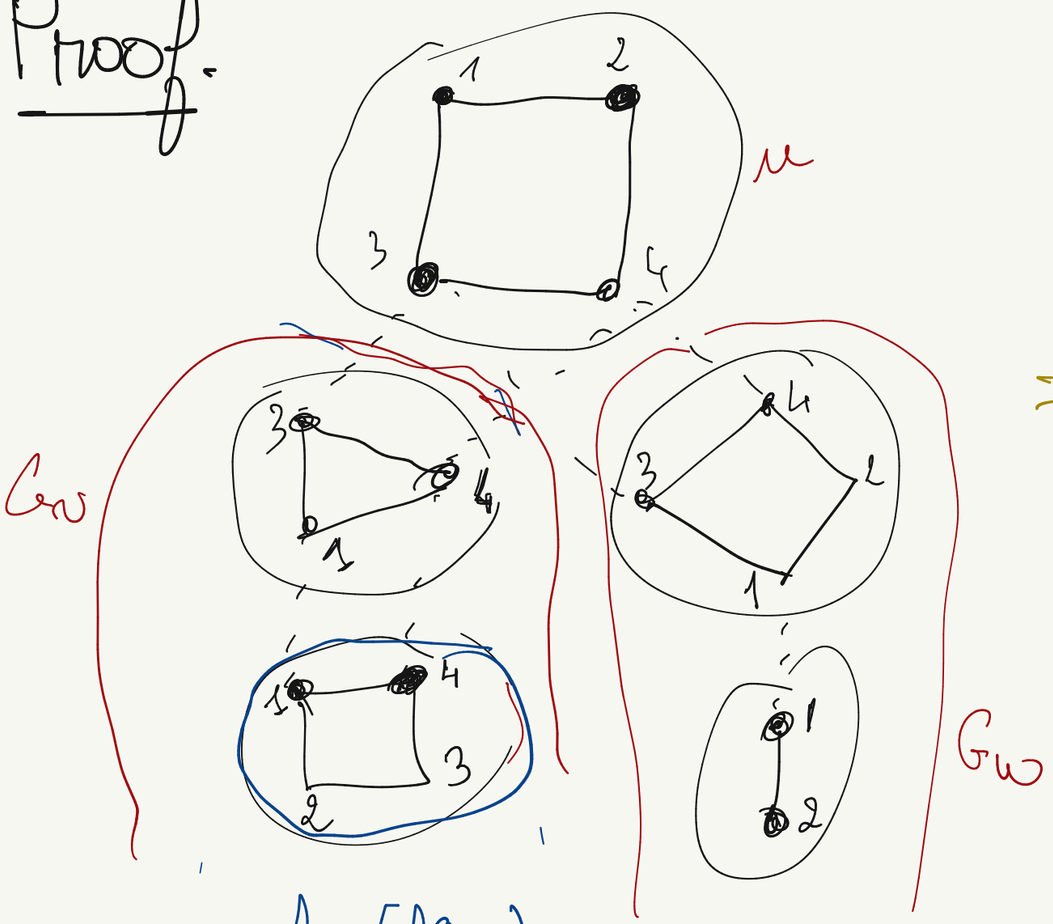
$val(ren_{i\to j}(t)) = ren_{i\to j}(val(t))$

- $t_1 // t_2$ is a term

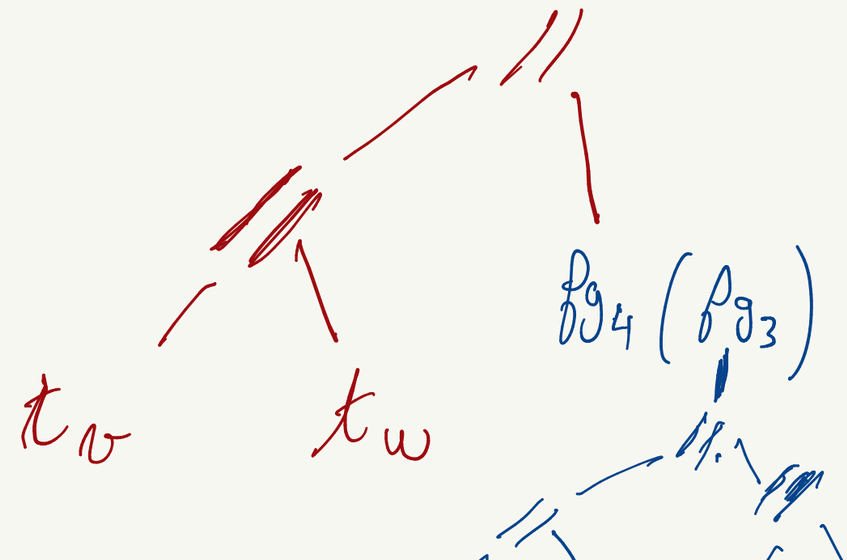
$val(t_1 // t_2) = val(t_1) // val(t_2)$

Theorem A: $Tw_D(G) \leq k \iff G \in T(HR_{k+1}, \{E\})$

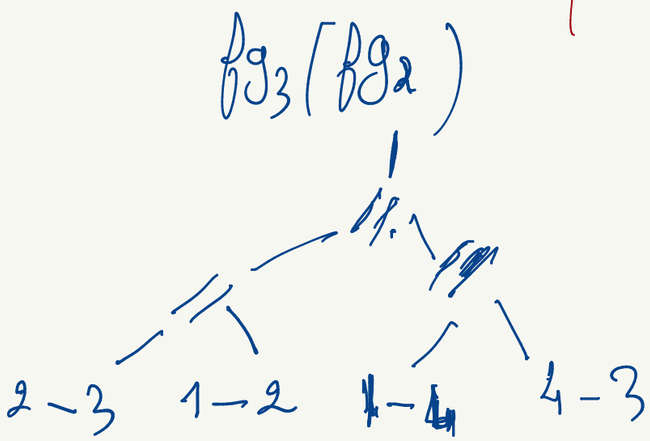
Proof.



$t_u =$

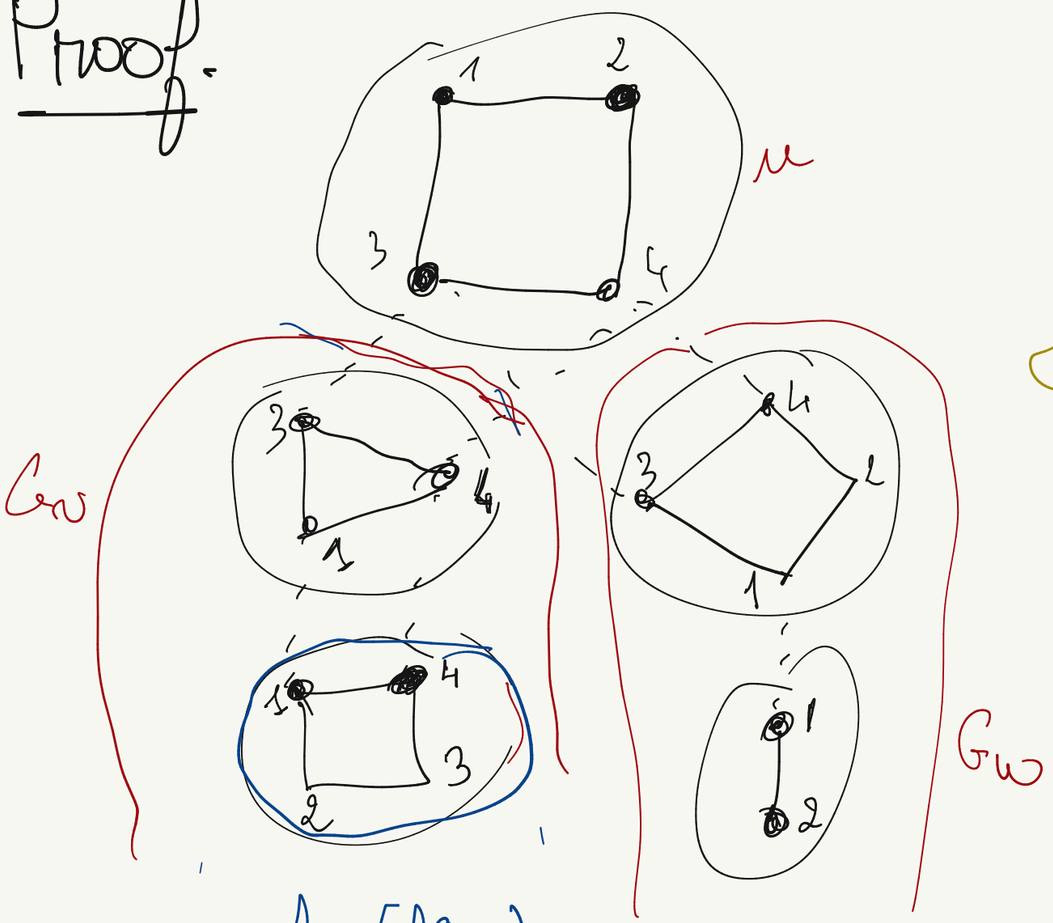


Create edges in bag u



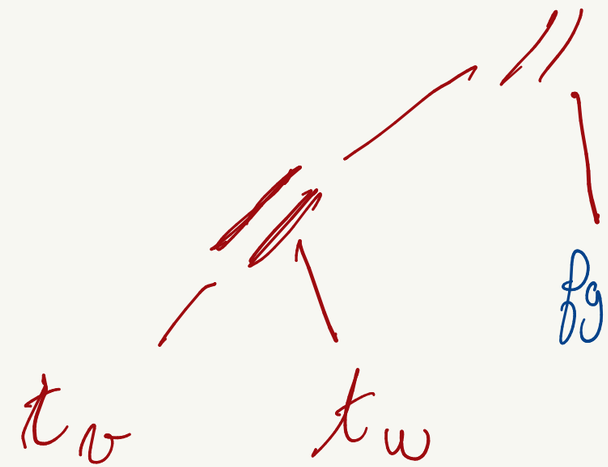
Theorem A: $\mathrm{Tw}(G) \leq k \iff G \in T(HR_{k+1}, \{E\})$

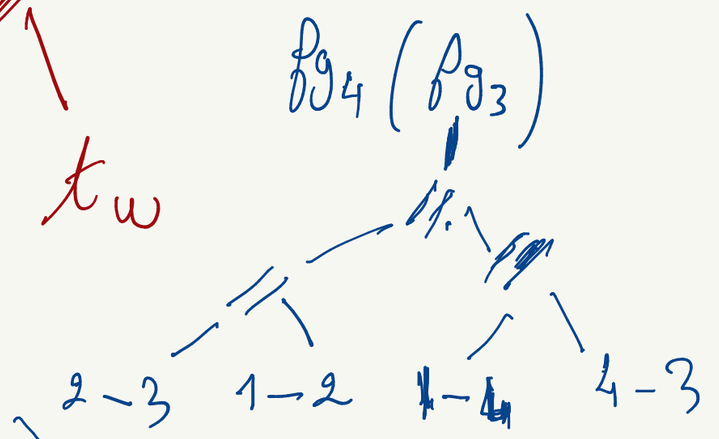
Proof.



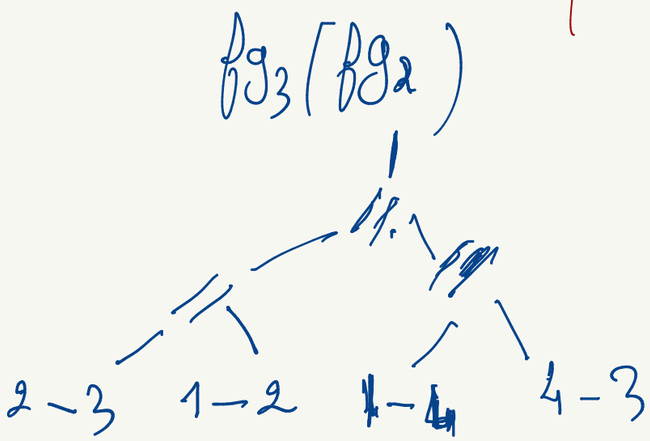
$\Longleftarrow$

$t_u =$





Create edges in bag u



Sources are bags

Combining Theorem A + Tree Automata + Bodlaender's Theorem

Theorem B (Courcelle '90) Every MSO_2 definable property can be solved in time $f(tw(G)) \cdot n$, for any n-vertex graph G.

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

MSO_2	MSO_1
• incidences : $inc(x, e)$	• adjacencies : $E(x, y)$
• $FO + \exists X,\ X \subseteq E \cup V$	• $FO + \exists X,\ X \subseteq V$

Combining Theorem A + Tree Automata + Bodlaender's Theorem

Theorem B (Courcelle '90) Every MSO_2 definable property can be solved in time $f(tw(G)) \cdot n$, 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 MSO_2 definable property can be solved in time $f(tw(G)) \cdot n$, for any n-vertex graph G.
[decision, search, optimise, count, list versions].

NATURAL QUESTION: IS BEST IN TERMS OF GRAPH CLASSES?

- YES if Language MSO_2 is wanted (Courcelle, Seese, ...)
- NO if we RESTRICT language ....

DETOUR TO CO-GRAPH

- Basic Graph : •
- 2 operations:
  - + disjoint union : ⊕
  - + complete join : ⊗

} val(terms) = co-graphs

DETOUR TO CO-GRAPHS

- Basic Graph : •
- 2 operations:
  - + disjoint union : ⊕
  - + complete join : ⊗

} val(terms) = co graphs

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

Theorem C [Folklore?]: Every MSO_1 definable property can be decided in linear time, on co-graphs.

of course HUGE HIDDEN CONSTANT

DETOUR TO CO-GRAPHS

- Basic Graph : •
- 2 operations:
  + disjoint union : ⊕
  + complete join : ⊗

} val(terms) = cographs

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

Theorem C [Folklore?]: Every MSO_1 definable property can be decided in linear time, on cographs.

of course HUGE CONSTANT

BASICS FOR CLIQUE-WIDTH OPERATIONS:

- Colour vertices
- add edges between colour classes

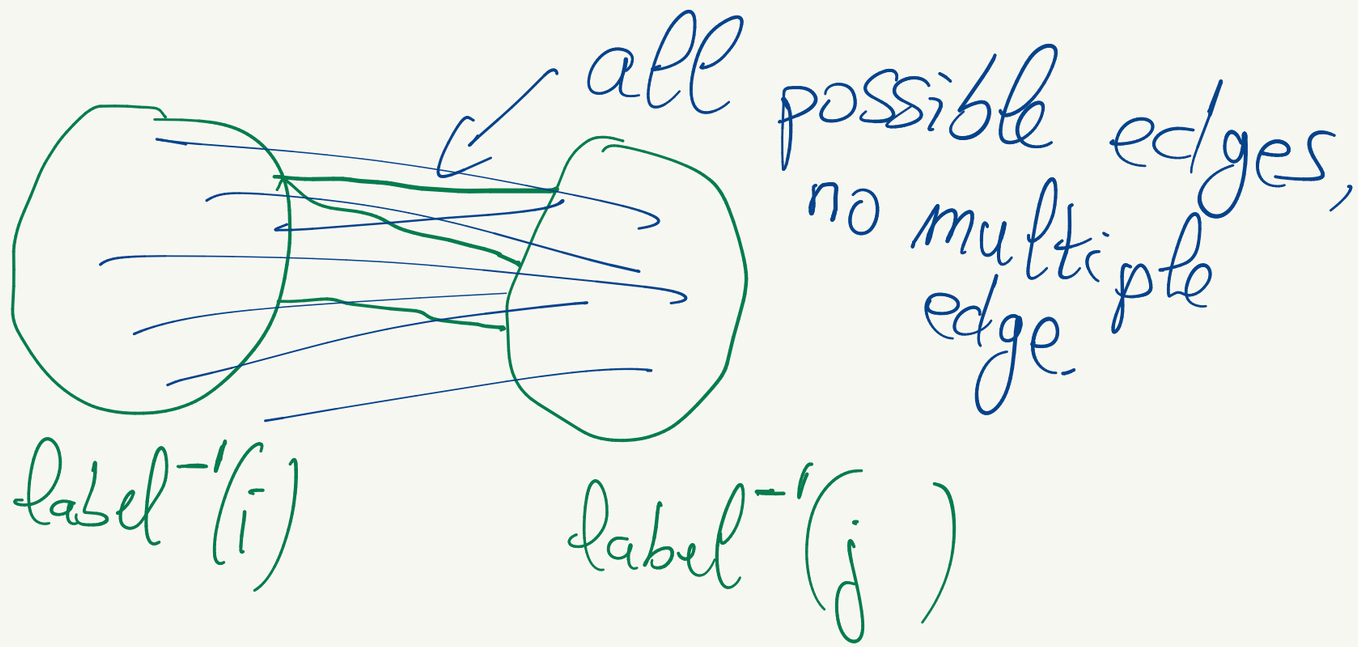
1. CLIQUE-WIDTH

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

- Basic Graph : 1

- 3 operations : $+$, $ren_{i \to j}$, $add_{i,j}$

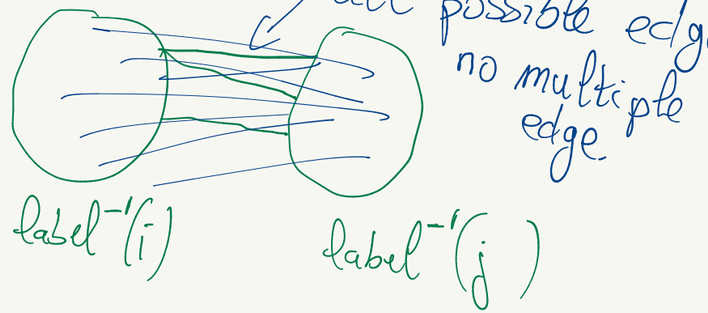
$add_{i,j}(b)$: $i \neq j$



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

- Basic Graph : $\bullet^1$
- 3 operations : $\oplus$, $ren_{i\to j}$, $add_{i,j} : i \neq j$

$add_{i,j}(G)$: $i \neq j$



- $val(1) : \bullet^1$
- $val(t \oplus t') : val(t) \oplus val(t')$
- $val(ren_{i\to j}(t)) : ren_{i\to j}(val(t))$
- $val(add_{i,j}(t)) : add_{i,j}(val(t))$

$$VR_k = \{\oplus, add_{i,j}, ren_{i\to j} : i,j \text{ in } [k]\}$$

- $val(t)$: k-labeled graph, $t \in T(VR_k, \{1\})$

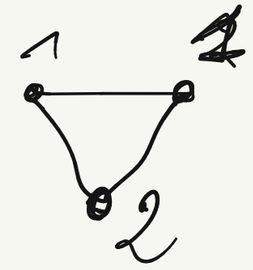
Examples

$i : ren_{1 \to i}(1)$

- $t_2 = add_{1,2}(1 \oplus 2)$
- $t_n = add_{1,2}(2 \oplus ren_{2 \to 1}(t_{n-1}))$

$val(t_2)$:

$val(t_3)$

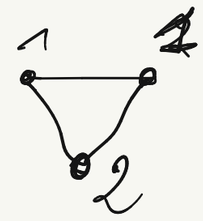


Examples

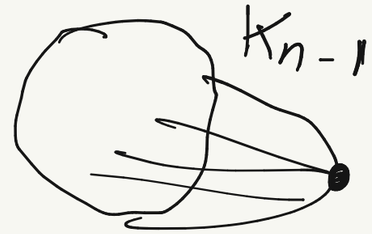
$i : ren_{1 \to i}(1)$

- $t_2 = add_{1,2}(1 \oplus 2)$ — $val(t_2)$:

- $t_n = add_{1,2}(2 \oplus ren_{2 \to 1}(t_{n-1}))$ — $val(t_3)$

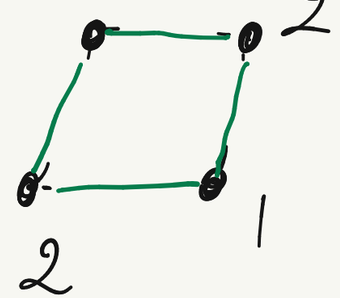


$val(t_n) =$

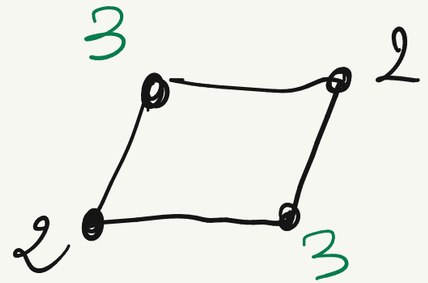


: K_n

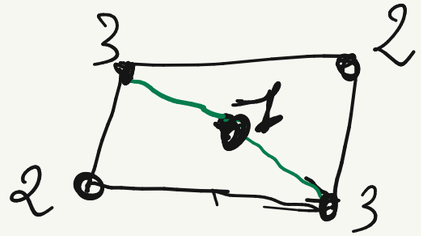
- $t' = add_{1,2}(1 \oplus 1 \oplus 2 \oplus 2)$ — $val(t')$:



- $t'' = ren_{1 \to 3}(t')$ — $val(t'')$

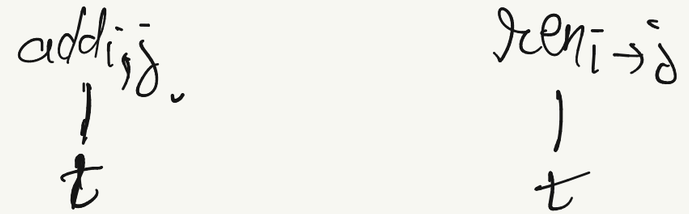


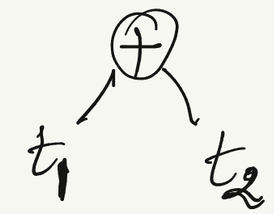
- $t''' = add_{1,3}(t' \oplus 1)$ — $val(t''')$



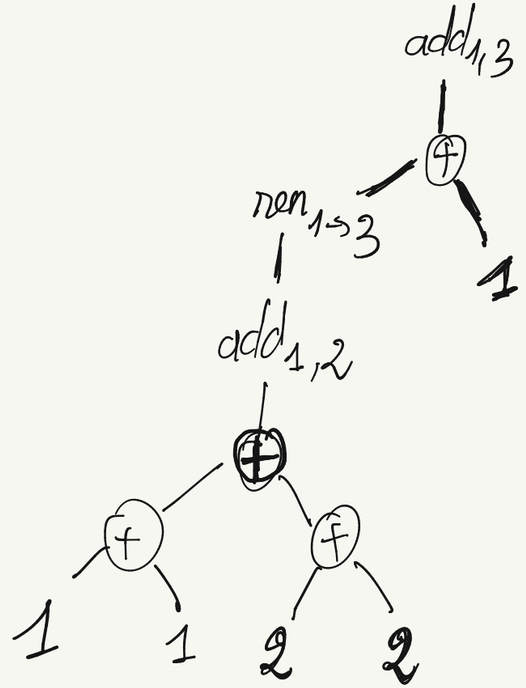
TERMS AS ROOTED LABELED TREES

$\rightarrow$ 1 : 

$\rightarrow$ $add_{i,j}(t)$ or $ren_{i\to j}(t)$: 

$\rightarrow$ $t_1 \oplus t_2$: 

Example:



The clique-width of a graph G, $cwd(G)$,
is the minimum k such that

$$G \cong val(t), \quad t \in T(VR_k, \{1\})$$

($\cong$: standard graph isomorphism)

DO NOT TAKE INTO ACCOUNT LABELS

The clique-width of a graph G, $cwd(G)$,
is the minimum k such that

$$G \approx val(t), \quad t \in T(VR_k, \{1\})$$

($\approx$: standard graph isomorphism)

When $G \approx val(t)$, we write sometimes
$i(x)$ in t to refer the vertex x of G
it is mapped to by $\approx$

Bound on Clique width

By definition, $cwd(G) \leq |V(G)|$. But,

Theorem 5 (Johansson '98) : $cwd(G) \leq n - k$ as long as $2^k \leq n - k$

Proof.

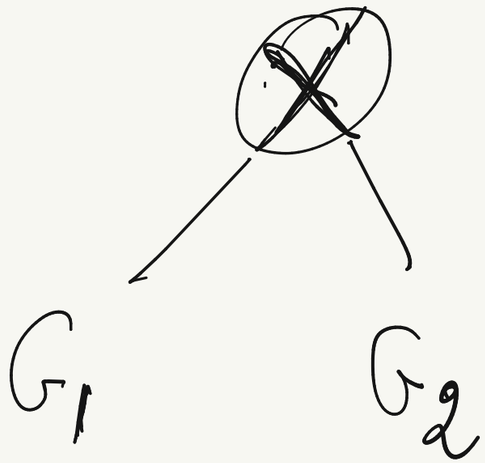
$\forall$ can be classified in at most 2^k classes wrt neighborhoods in V_2

- use $n-k$ labels to construct $G[V_1]$ and relabel wrt neighbors in V_2 $\leq 2^k$ labels
- add 1 by 1 vertices in V_2 : one extra label.

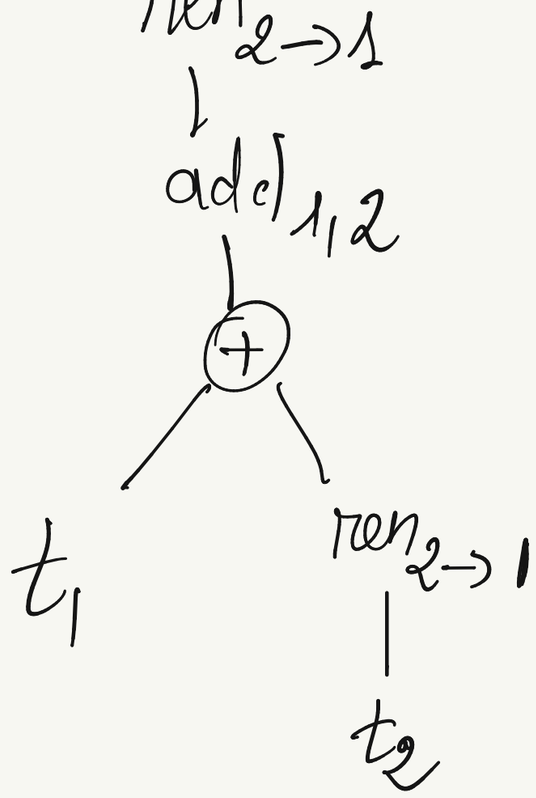
2. Some Graph Classes of (un) bounded clique-width

Proposition 1 : G cograph $\Leftrightarrow$ $G \simeq val(t)$, $t \in T(VR_2, \{1\})$.

Proof.

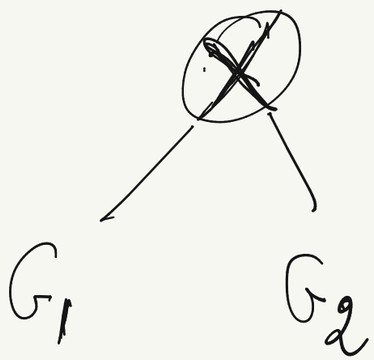


$=$

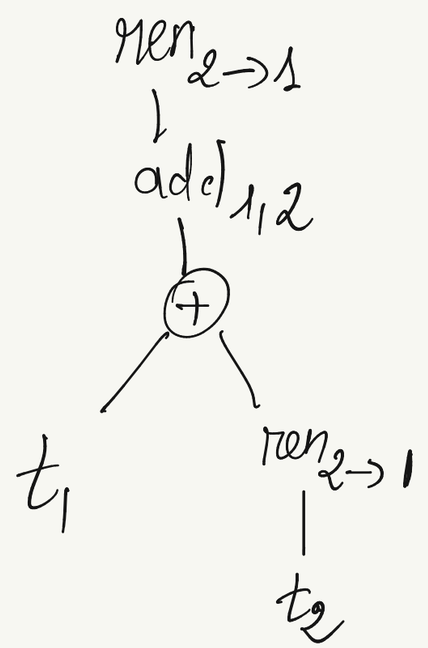


Proposition 1 : G cograph $\Longleftrightarrow$ $G \simeq val(t)$, $t \in T(VR_2, \{1\})$.

Proof.



$=$



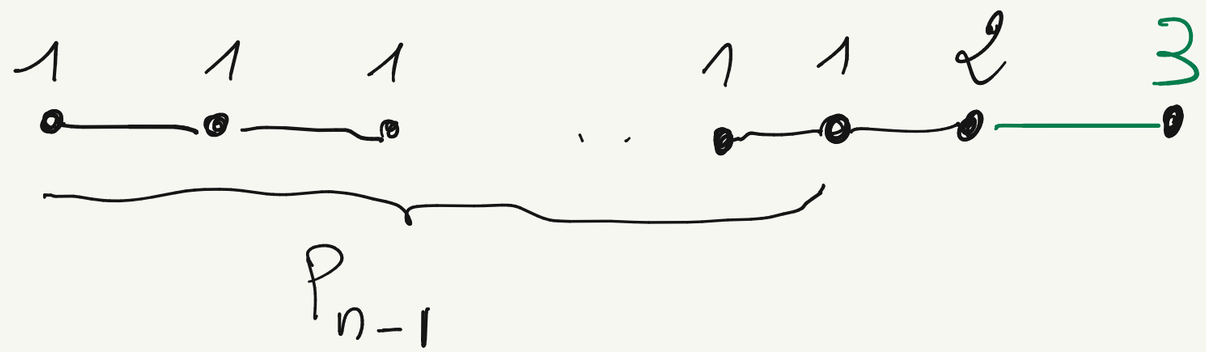
The other direction needs a normalisation : when using $add_{1,2}$, no edges between vertices labeled 1 and those labeled 2.

Paths.

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

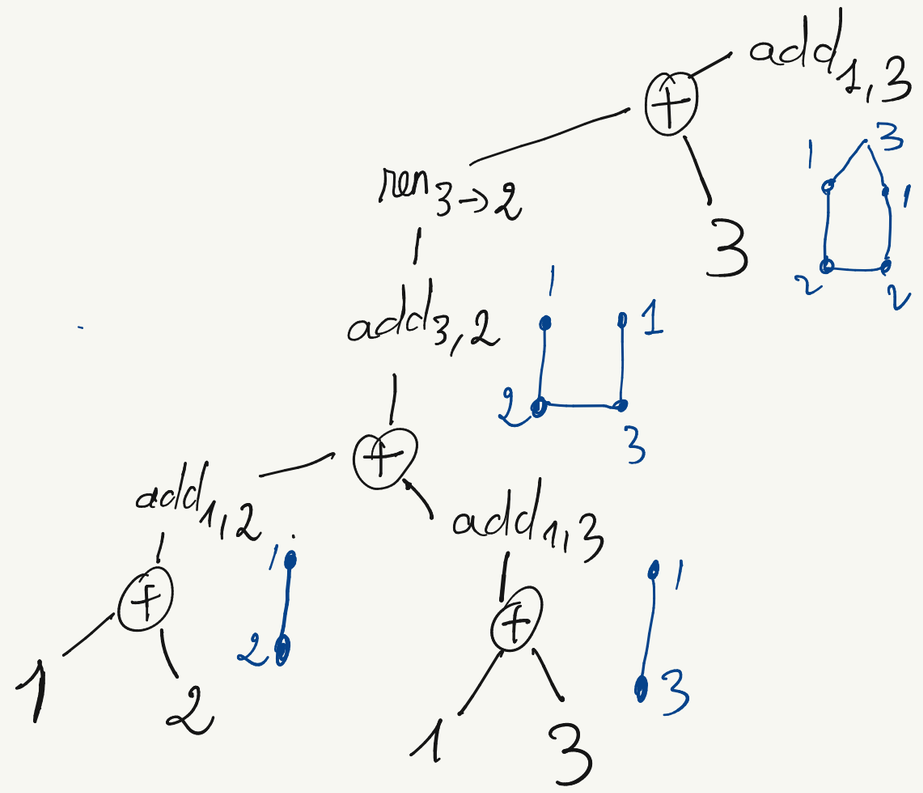
$$t_4 = add_{2,3}\Big(ren_{3\to 2}\big(ren_{2\to 1}\big(add_{1,3}\big(add_{1,2}(1 \oplus 2) \oplus 3\big)\big)\big) \oplus 3\Big)$$

$$t_{n+1} = add_{2,3}\Big(ren_{3\to 2}\big(ren_{2\to 1}(t_n)\big) \oplus 3\Big)$$



Cycles

- $cwd(C_4) = 2$ (ω-graph)
- $cwd(C_5) = 3$



- $cwd(C_n) \leq 4$

Distance-HEREDITARY GRAPHS

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

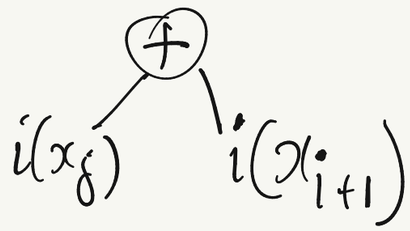
Proposition 2.1 [Golumbic, Rotics '00]: $cwd(DH) \leq 3$.

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

$\perp$ is never used to create an edge.

- t_i generates $G[x_1, .., x_i]$, x_{i+1} is twin/pendant of x_j
- $t_{i+1} = t_i[i(x_j)/t']$ where t' is

t':



false twins

Distance-HEREDITARY GRAPHS

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

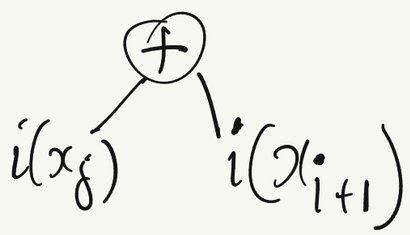
Proposition 2.1 [Golumbic, Rotics '00]: $cwd(DH) \leq 3$.

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

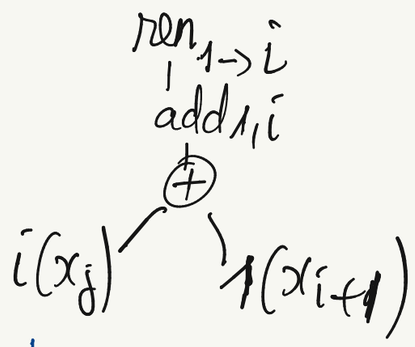
$\perp$ is never used to create an edge.

- t_i generates $G[x_1, .., x_i]$, x_{i+1} is twin/pendant of x_j
- $t_{i+1} = t_i[i(x_j)/t']$ where t' is

t':



false twins



true twins

Distance-HEREDITARY GRAPHS

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

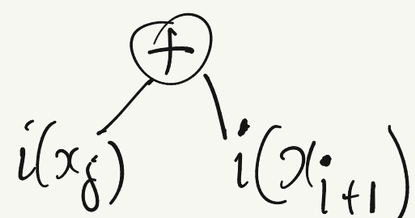
Proposition 2.1 (Golumbic, Rotics '00): $cwd(DH) \leq 3$.

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

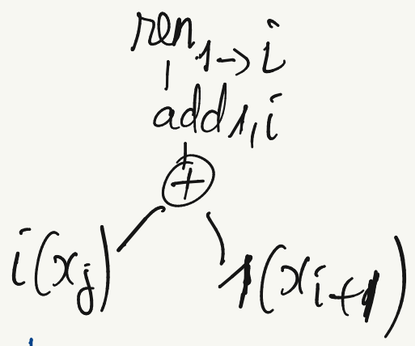
1 is never used to create an edge.

- t_i generates $G[x_1, \dots, x_i]$, x_{i+1} is twin/pendant of x_j
- $t_{i+1} = t_i[t'/i(x_j)]$ where t' is

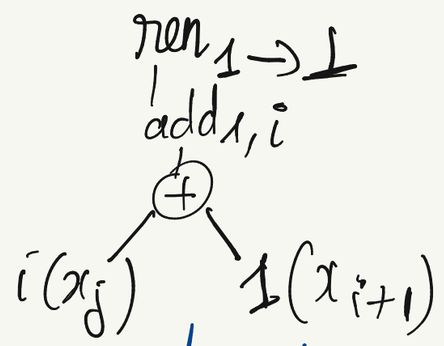
t':



false twins



true twins



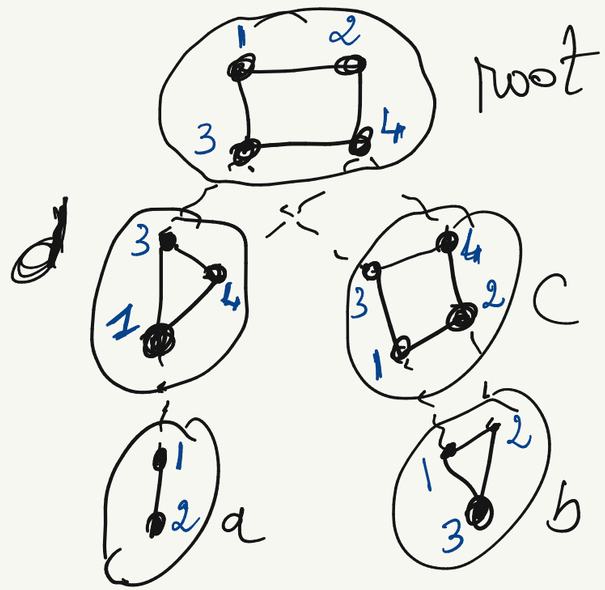
pendant

Graphs of treewidth k

Proposition 2.2 : $cwd(G) \leq 2^{tw(G)+1}$

Proof: • (T, f) a tree-decomposition of width tw

• Do a proper $(tw+1)$-coloring of G

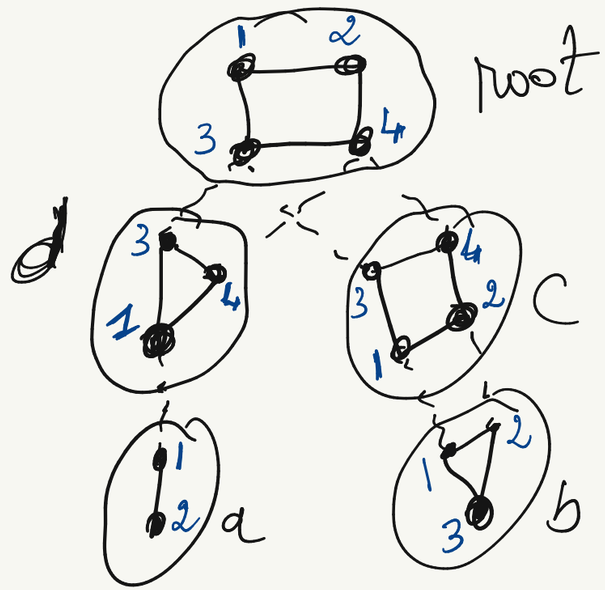


Graphs of Treewidth k

Proposition 2.2 : $cwd(G) \leq 2^{tw(G)+1}$

Proof:

- (T, f) a tree-decomposition of width tw
- Do a proper $(tw+1)$-coloring of G
- If $u \in V(T)$, every edge between $V_u = \bigcup_{v \leq_T u} f(v)$ and $V(G) \setminus V_u \subseteq G[f(u)]$
- $\forall u$, compute t_u, for $G[V_u \setminus f(parent(u))]$ such that each vertex has label $\{ color(y) : y \in f(u) \cap f(parent(u)), xy \in E(G) \}$



Graphs of Treewidth k

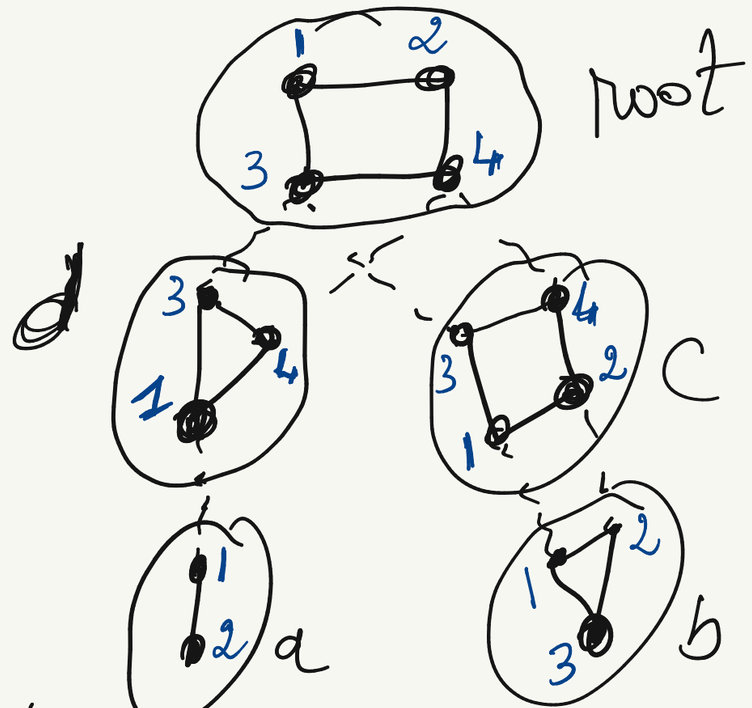
Proposition 2.2 : $cwd(G) \leq 2^{tw(G)+1}$

Proof: • (T, f) a tree-decomposition of width tw

• Do a proper $(tw+1)$-coloring of G

• If $u \in V(T)$, every edge between $V_u := \bigcup_{v \leq_T u} f(v)$ and $V(G) \setminus V_u \subseteq G[f(u)]$

• $\forall u$, compute t_u ; for $G[V_u \mid f(parent(u))]$ such that each vertex has label $\{ color(y) : y \in f(u) \cap f(parent(u)), xy \in E(G) \}$



$$t_a = ren_{2 \to 1}(2)$$

$$t_d = ren_{1 \to \{3,4\}}\left(ren_{2 \to \emptyset}\left(add_{1,2}(1 \oplus t_a)\right)\right)$$

$$t_b = ren_{3 \to \{1,2\}}(3)$$

$$t_c = ren_{2 \to \{3,4\}}\left(ren_{1 \to \{3,4\}}\left(add_{\{1,2\},2}\left(add_{\{1,2\},1}\left((add_{1,2}\left((t_d \oplus 1 \oplus 2)\right)\right)\right)\right)\right)\right)$$

Graphs of Treewidth k

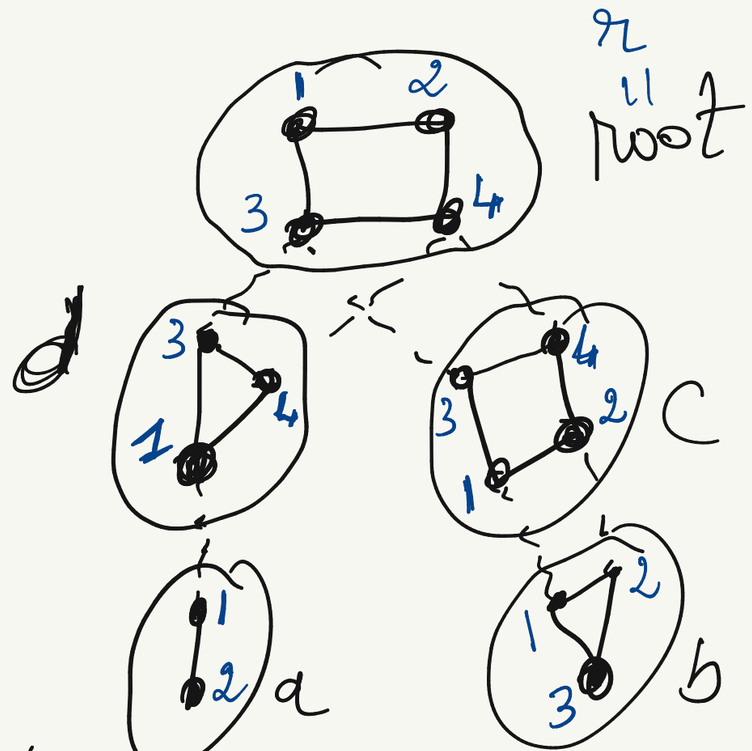
Proposition 2.2 : $cwd(G) \leq 2^{tw(G)+1}$

Proof: • (T, f) a tree-decomposition of width tw

• Do a proper $(tw+1)$-coloring of G

• If $u \in V(T)$, every edge between $V_u := \bigcup_{v \leq_T u} f(v)$ and $V(G) \setminus V_u \subseteq f(u)$.

• $\forall u$, compute t_u, for $G[V_u \mid f(\text{parent}(u))]$ such that each vertex has label $\{ \text{color}(y) : y \in f(u) \cap f(\text{parent}(u)), xy \in E(G) \}$



$t_a = ren_{2 \to 1}(2)$

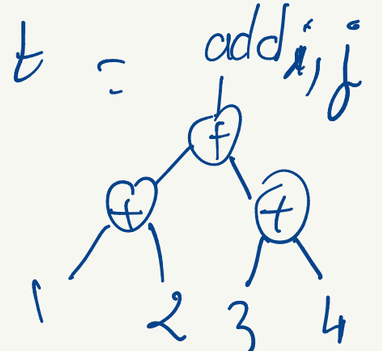
$t_d = ren_{1 \to \{3,4\}}(ren_{2 \to \emptyset}(add_{1,2}(1 \oplus t_a)))$

$t_b = ren_{3 \to \{1,2\}}(3)$

$t_c = ren_{2 \to \{3,4\}}(ren_{1 \to \{3,4\}}(add_{\{1,2\},2}(add_{\{1,2\},1}((add_{1,2}((t_d \oplus 1 \oplus 2)))))))$

$t_r = add_{\{3,4\},4}(add_{\{3,4\},3}(t_d \oplus t_c \oplus t))$

$t = add_{i,j}$



Graphs of Treewidth k

Proposition 2.2: $\forall k.\ \exists G,\ tw(G)=k,\ cwd(G) \geq 2^{\lfloor k/2 \rfloor - 1}$

~~$cwd(G) \leq 2^{tw(G)+1}$~~ $cwd(G) \leq 3 \cdot 2^{tw(G)-1}$

Proposition 2.3: $cwd(Line(G)) \leq f(tw(G))$

Proposition 2.4: On $K_{p,p}$-subgraph free graphs,

$f(tw(G)) \leq cwd(G) \leq g(tw(G))$

Some Operations on Graphs

- $H \subseteq_i G$: induced sub graph
  $cwd(H) \leq cwd(G)$

Some Operations on Graphs

- $H \subseteq_i G$: induced subgraph
  $cwd(H) \leq cwd(G)$
- $\overline{G}$ = complement of G.
  $cwd(\overline{G}) \leq 2 \cdot cwd(G)$
  → any term in $T(VR_k, \{1\})$ is equivalent to
  Some using binary $\oplus_{R,f}$: $R \subseteq [k] \times [k]$, $f : [k] \to [k]$
  A relabeling function
  edges added between operands by R
  → easy now to construct $\overline{G}$: replace $\oplus_{R,f}$ by $\oplus_{\overline{R},f}$
  → any term with $\oplus_{R,f}$ translates into one in $T(VR_{2k}, \{1\})$

Some Operations on Graphs

- $H \subseteq_i G$: induced subgraph
  $cwd(H) \leq cwd(G)$

- $\bar{G}$ = complement of G.
  $cwd(\bar{G}) \leq 2 \cdot cwd(G)$

- Substitution : $v \in V(G)$, $V(G) \cap V(H) = \emptyset$ $\quad G' = G[H/v]$
  - $V(G') = V(G) \uplus V(H) \setminus \{v\}$
  - $E(G') = E(G \setminus v) \uplus E(H) \uplus N_G(v) \times V(H)$

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

Some Operations on Graphs

- $H \subseteq_i G$: induced sub graph
  $cwd(H) \leq cwd(G)$

- $\bar{G}$ = complement of G.
  $cwd(\bar{G}) \leq 2 \cdot cwd(G)$

- Substitution : $v \in V(G)$, $V(G) \cap V(H) = \emptyset$ $\quad G' = G[H/v]$
  - $V(G') = V(G) \uplus V(H) \setminus \{v\}$
  - $E(G') = E(G \setminus v) \uplus E(H) \uplus N_G(v) \times V(H)$

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

Observation 2.1 $cwd(G') = \max\{cwd(G), cwd(H)\}$

$val(t_G) \simeq G$, $val(t_H) \simeq H$

$$val\left(t_G\left[\left(\bigodot_{neni \to l} (t_H)\right) / l(v)\right]\right) \simeq G'$$

Some Operations on Graphs

- $H \subseteq_i G$: induced sub graph
  $cwd(H) \leq cwd(G)$

- $\bar{G}$ = complement of G.
  $cwd(\bar{G}) \leq 2 \cdot cwd(G)$

- Substitution : $v \in V(G)$, $V(G) \cap V(H) = \emptyset$ $\quad G' = G[H/v]$
  - $V(G') = V(G) \uplus V(H) \setminus \{v\}$
  - $E(G') = E(G \setminus v) \uplus E(H) \uplus N_G(v) \times V(H)$

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

Observation 2.1 $cwd(G') = \max\{cwd(G), cwd(H)\}$

$val(t_G) \simeq G$, $val(t_H) \simeq H$

$$val\left(t_G\left[\left(\bigodot_{neni \to \ell} (t_H)\right) / \ell(v)\right]\right) \simeq G'$$

Theorem 2.1 : For every graph G

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

↑ every module is either all or a singleton.

Corollary of Theorem 2.1

Theorem 2.1: For every graph G

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

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

Examples: (P_5, house)-free, chordal $(\cdot \text{---} \cdot \text{---} \cdot \text{---} \cdot)$-free, $(\text{diamond}, \text{2K}_1)$-free, $(\text{house}, \cdot \text{---} \cdot \text{---} \cdot \text{---} \cdot)$-free, $(\text{2K}_1, \text{house})$-free, $(P_5, \text{diamond})$-free, ...

complete classification for those excluding a one-vertex extension of P_4, graphs of size ≤ 4

non perfect $(4K_1, C_4, C_5)$-free, ...

UNBOUNDED CLIQUE-WIDTH

(A) SUPER — LOGIC

Some Graph classes have undecidable MS_1-Theory.
Yet, others can encode some with undecidable theories.

- If $cwd(G) \leq k \iff G \approx val(t)$, $t \in T(VR_k, \mathcal{H})$
- t a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

$\Longrightarrow$ 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 MS_1-Theory.

Yet, others can encode some with undecidable theories.

- If $cwd(G) \leq k \iff G \simeq val(t)$, $t \in T(VR_k, \{1\})$
- t a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

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

> For every MSO_1 formula φ, there is a deterministic tree-automata $\mathcal{A}_\varphi$ such that
> $G \models \varphi \iff t \in L(\mathcal{A}_\varphi)$

UNBOUNDED CLIQUE-WIDTH

(A) SUPER — LOGIC

Some Graph classes have undecidable MS_1-Theory.
Yet, others can encode some with undecidable theories.

- If $cwd(G) \leq k \iff G \simeq val(t)$, $t \in T(\overline{VR_k}, \{1\})$
- t a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

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

For every MSO_1 formula φ, there is a deterministic tree-automata $\mathcal{A}_\varphi$ such that
$G \models \varphi \iff t \in L(\mathcal{A}_\varphi)$

$\Downarrow\Downarrow$

MSO_1 decidable on graphs of clique-width k.

UNBOUNDED CLIQUE-WIDTH

(A) SUPER-LOGIC

Some Graph classes have undecidable MS_1-Theory.

Yet, others can encode some with undecidable theories.

- If $cwd(G) \leq k \iff G \simeq val(t)$, $t \in T(VR_k, \mathcal{F}_k)$
- t a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

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

> For every MSO_1 formula φ, there is a deterministic tree-automata $\mathcal{A}_\varphi$ such that
> $G \models \varphi \iff t \in L(\mathcal{A}_\varphi)$

$\Downarrow\Downarrow$

> MSO_1 decidable on graphs of clique-width k.

Examples:

- All graphs
- Grids
- planar graphs of degree ≤ 3
- Split graphs
- Bipartite graphs

...

UNBOUNDED CLIQUE-WIDTH

(A) SUPER-LOGIC

Some Graph classes have undecidable MS_1-Theory.
Yet, some can encode some with undecidable theories.

- If $cwd(G) \leq k \iff G \simeq val(t)$, $t \in T(VR_k, \{1\})$
- t a rooted labeled tree
- A bottom-up tree-automata to decide whether $xy \in E(G)$

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

> For every MSO_1 formula φ, there is a deterministic tree-automata $\mathcal{A}_\varphi$ such that
> $G \models \varphi \iff t \in L(\mathcal{A}_\varphi)$

$\Downarrow\Downarrow$

> MSO_1 decidable on graphs of clique-width k.

Examples:

- All graphs
- Grids
- planar graphs of degree ≤ 3
- Split graphs
- Bipartite graphs

} encoding

But bounds are bad

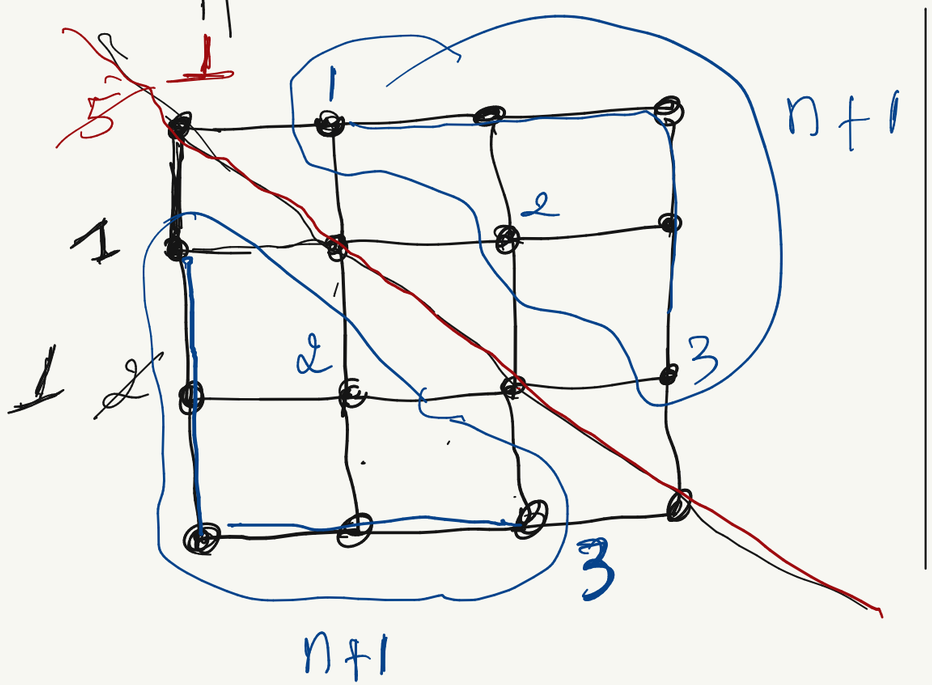
UNBOUNDED CLIQUE-WIDTH

(B) Use Hands and Case analysis

B.1: Square Grids

$$cwd(G_{n,n}) = n+1$$

Upper Bound



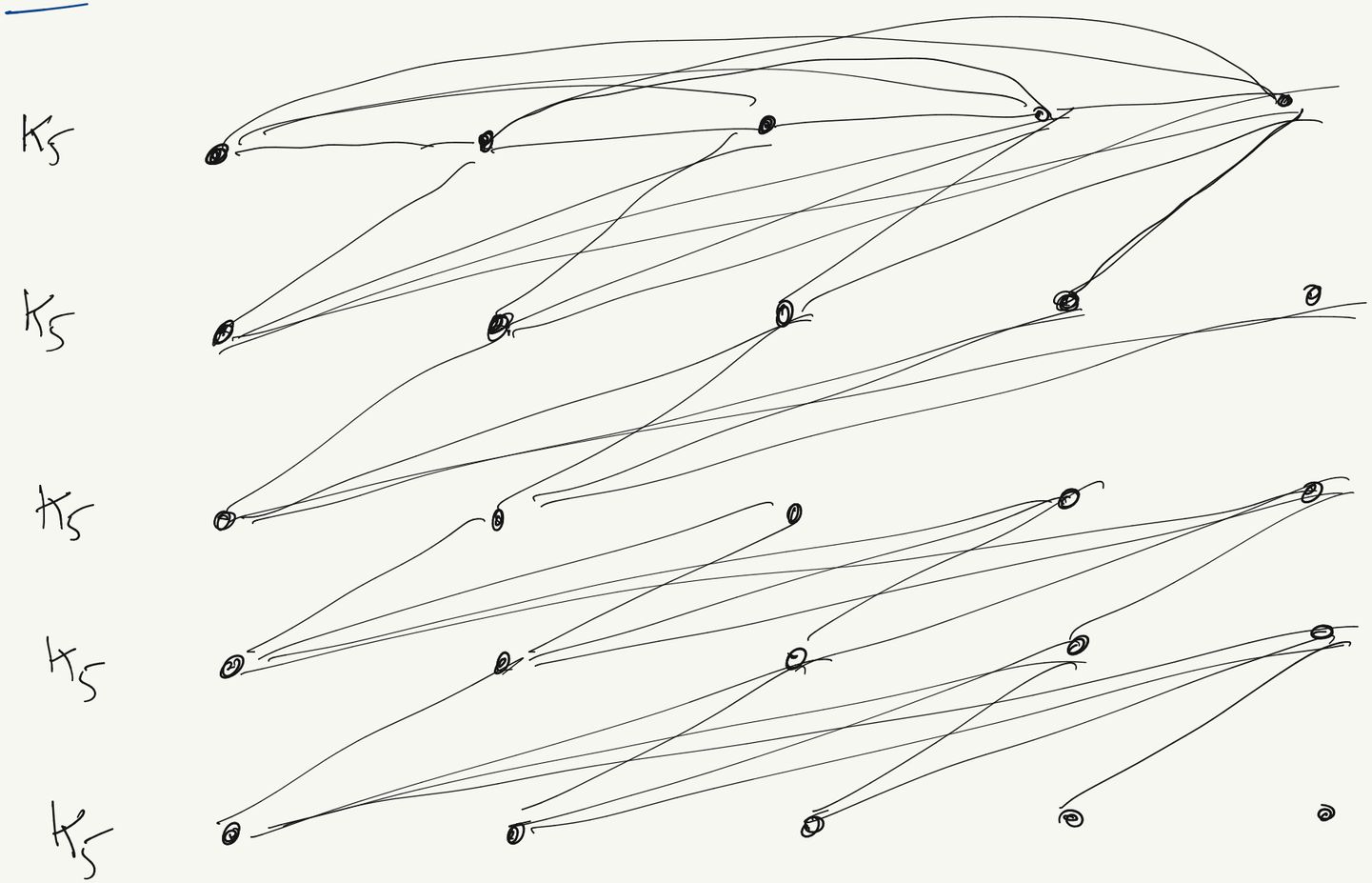
Lower bound

- 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



universal unit-interval graph

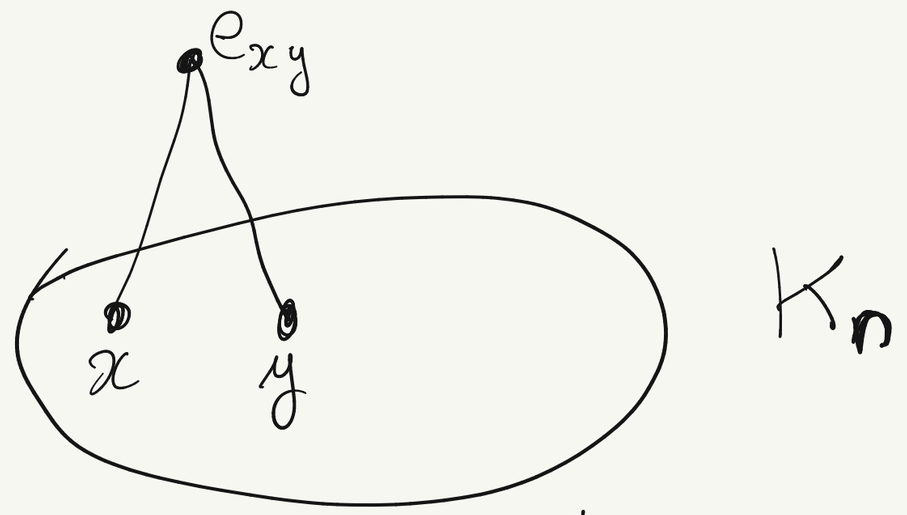
$cwd = n+1$

UNBOUNDED CLIQUE-WIDTH

(B) Use Hands and Case analysis

B.3. Split graphs

$G \longrightarrow$



$cwd \geq \frac{n}{72}$

- for any term, there is node u. such that

$$\frac{n(n+1)}{6} \leq |V(G_u)| \leq \frac{n(n+1)}{3}$$

look at the edges in the other side incident with it: counting will contradict that $cwd < n/72$.

Some difficulties to deal with Clique-width

- $\subseteq_i$: the only monotone operation.
  - → The audience knows how hard it can be to study structure with $\subseteq_i$
  - → few graph classes are wqo under $\subseteq_i$
- 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 : $cwd(G) \leq 3$
  - → One can compute exactly for Bounded treewidth.

3. Detour to Some Algorithmic Applications

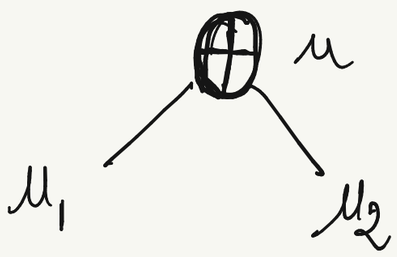
- Courcelle et al. Theorem give intractable FPT algorithm :
- Your own DP if you want a better one.

ASSUME a term is given

3.1. Independent Set

- Do a bottom-up traversal and compute for u and $\lambda \subseteq [k]$

$$tab_u[\lambda] = \max\{X : tab_G(X) = \lambda\}$$

- 

$$: tab_u[\lambda] = \max\{tab_{u_1}[\lambda], tab_{u_2}[\lambda]\}$$

- 

u :
$$tab_u[\lambda] = \begin{cases} tab_u[\lambda] & \text{if } j \notin \lambda \\ \max\{tab_{u_1}[\lambda], tab_{u_1}[\lambda \cup i]\} \end{cases}$$

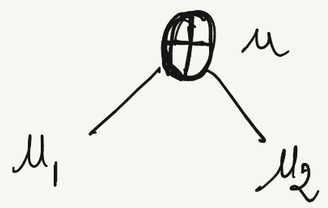
- 

u :
$$tab_u[\lambda] = \begin{cases} \emptyset & \text{if } \{i,j\} \subseteq \lambda \\ tab_{u_1}[\lambda] & \text{otherwise.} \end{cases}$$

3.1. Independent Set

- Do a bottom-up traversal and compute for u and $\lambda \subseteq [k]$

$$tab_u[\lambda] = \max\{X : tab_G(X) = \lambda\}$$

- 

$$tab_u[\lambda] = \max\{tab_{u_1}[\lambda], tab_{u_2}[\lambda]\}$$

- 

$$tab_u[\lambda] = \begin{cases} tab_u[\lambda] & \text{if } j \notin \lambda \\ \max\{tab_{u_1}[\lambda], tab_{u_1}[\lambda \cup i]\} \end{cases}$$

- 

$$tab_u[\lambda] = \begin{cases} \emptyset & \text{if } \{i,j\} \subseteq \lambda \\ tab_{u_1}[\lambda] & \text{otherwise.} \end{cases}$$

time: $= 2^{k+1} \cdot n$ if t given.
$= 2^k$: optimal under ETH

3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each coloring.
- Assume you want to check whether G d-colorable.
- For each node u, keep the following for G_u,
  for each proper d-coloring $(X_1, X_2, \dots, X_d)$
  $(C_1, C_2, \dots, C_k)$: $C_i = \{j \in [d] : i \in lab(X_j)\}$
- At most $d^{d \cdot k}$ possible entries
- We update similarly as previous algorithm.

3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each coloring.
- Assume you want to check whether G 2-colorable.
- For each node u, keep the following for G_u,
  for each proper 2-coloring $(X_1, X_2, \ldots, X_2)$
  $(C_1, C_2, \ldots, C_k)$: $C_i = \{j \in [2] : i \in lab(X_j)\}$
- At most 2^{2-k} possible entries
- We update similarly as previous algorithm

- A clever one with $(2^2 - 2)^k$ possible entries.
  no $(2^2 - 2 - \varepsilon)^k$ one under SETH
  $\forall \varepsilon > 0$

3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each coloring.
- For a proper coloring $\chi = (X_1, X_2, \dots, X_k)$, associate the following function:

$$\begin{aligned} [k]^k &\longrightarrow [k] \\ L &\longmapsto \{j : lab(X_j) = L\} \end{aligned}$$

3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each coloring.
- For a proper coloring $\chi = (\chi_1, \chi_2, \dots, \chi_n)$, associate the following function:

$$\begin{aligned} [n]^k &\longrightarrow [\alpha] \\ L &\longmapsto \{j : lab(\chi_j) = L\} \end{aligned}$$

- $\alpha \leq n \implies n^{2^k}$ possible such functions.
- The update is the same as previously.

3.2. Chromatic Number

- As for the previous algorithm, keep the labels for each coloring.
- For a proper coloring $\lambda = (X_1, X_2, \dots, X_\alpha)$, associate the following function:

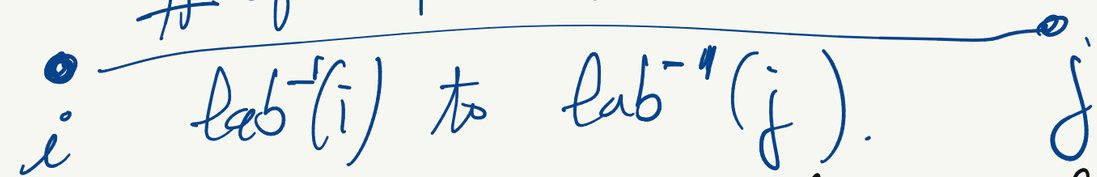
$$[2^k] \longrightarrow [\alpha]$$
$$L \longmapsto \{j : lab(X_j) = L\}$$

- $\alpha \leq n \implies n^{2^k}$ possible such functions.
- The update is the same as previously.

- Optimal under ETH: no $n^{2^{o(k)}}$

3.3 HAMILTONIAN PATH

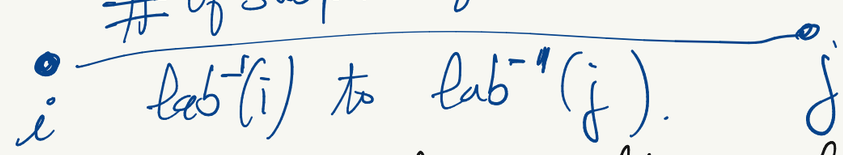
- If P is a HAM PATH, and u a node, $P \cap V(G_u)$ is a collection of paths.
- Construct the multigraph $\mathcal{M}(P)$ on $[k]$



- n^{k^2} possible such multigraphs.
- $P \cup Q$ a HAM Path $\Longleftrightarrow$ an alternating eulerian tour on $\mathcal{M}(P) \cup \mathcal{M}(Q)$.

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 $\mathcal{M}(P)$ on $[k]$



- n^{k^2} possible such multigraphs.
- $P \cup Q$ a HAM Path $\Leftrightarrow$ an alternating eulerian tour on $\mathcal{M}(P) \cup \mathcal{M}(Q)$.
- $P_1 \simeq P_2$ if $\mathcal{M}(P_1)$ and $\mathcal{M}(P_2)$
  - same connected components
  - same degree sequence.

} $n^{O(k^2)}$ time algorithm

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 $\mathcal{M}(P)$ on $[k]$

- n^{k^2} possible such multigraphs.
- $P \cup Q$ a HAM Path $\Leftrightarrow$ an alternating eulerian tour on $\mathcal{M}(P) \cup \mathcal{M}(Q)$.
- $P_1 \simeq P_2$ if $\mathcal{M}(P_1)$ and $\mathcal{M}(P_2)$
  - same connected components
  - same degree sequence.

  } $n^{O(k)}$ time algorithm

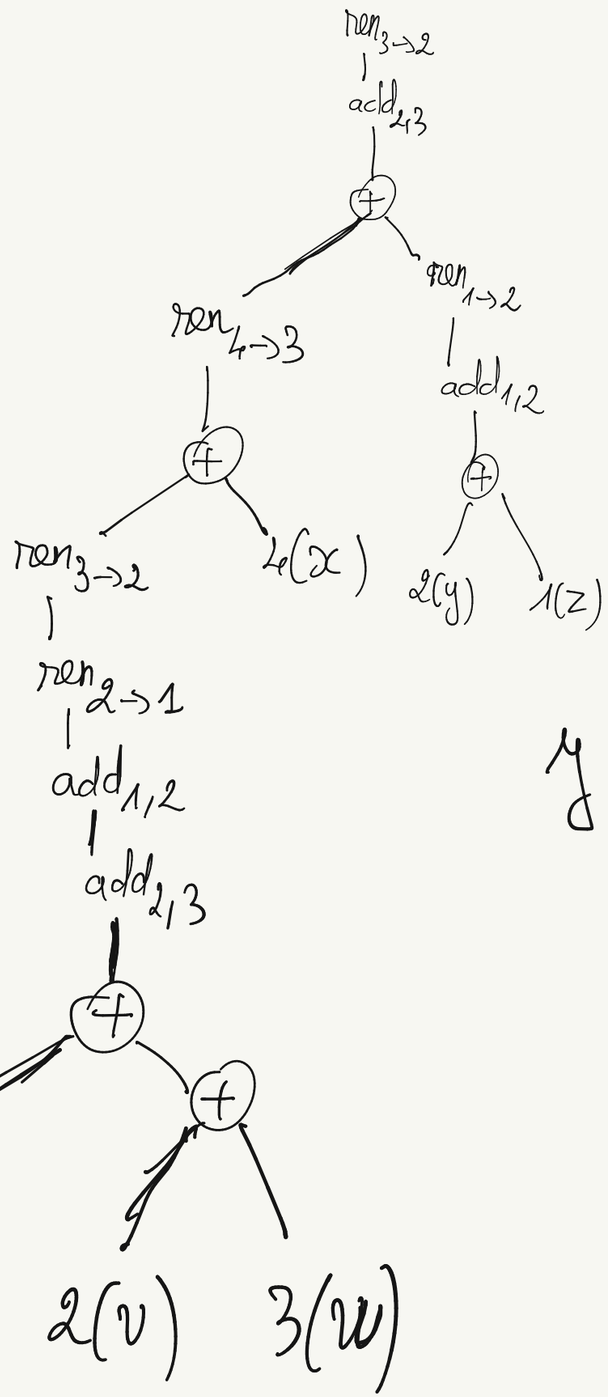
- $n^k \cdot k!$ possible entries. no $n^{o(k)}$ under ETH.

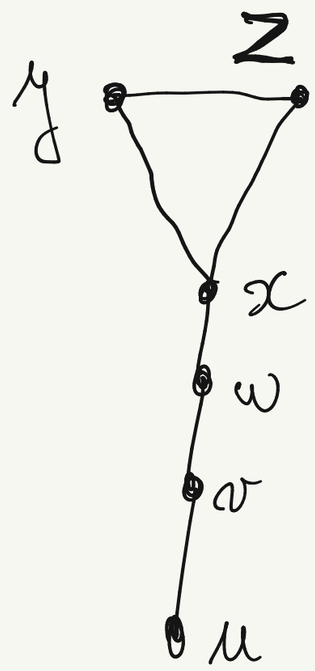
Many other examples

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

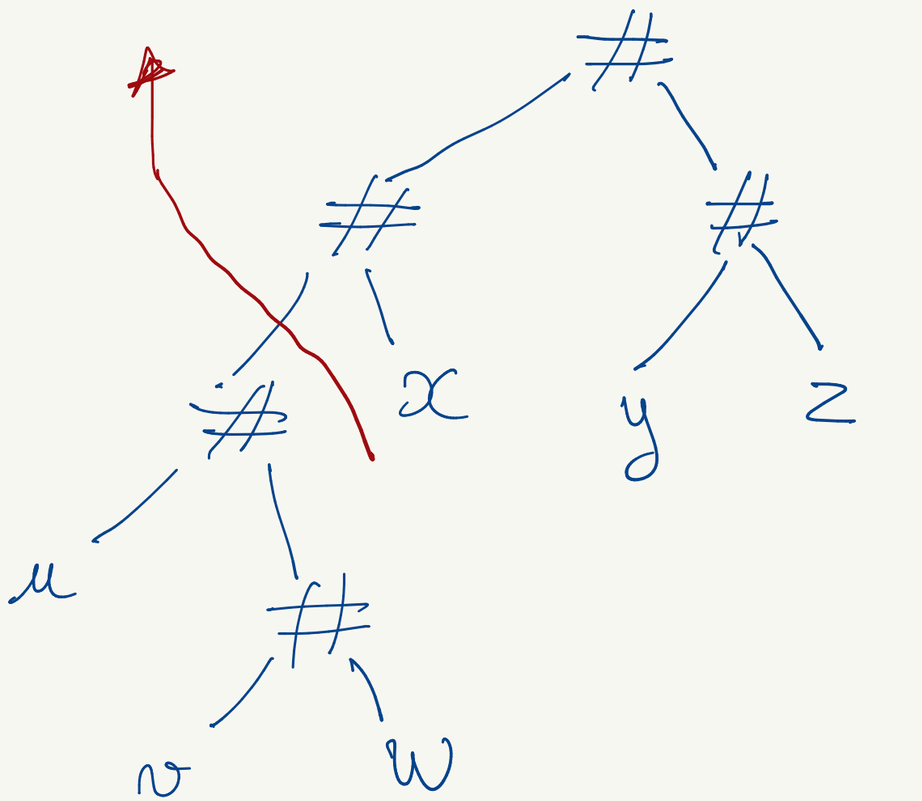
4. Some equivalent measures

Look at Neighborhoods



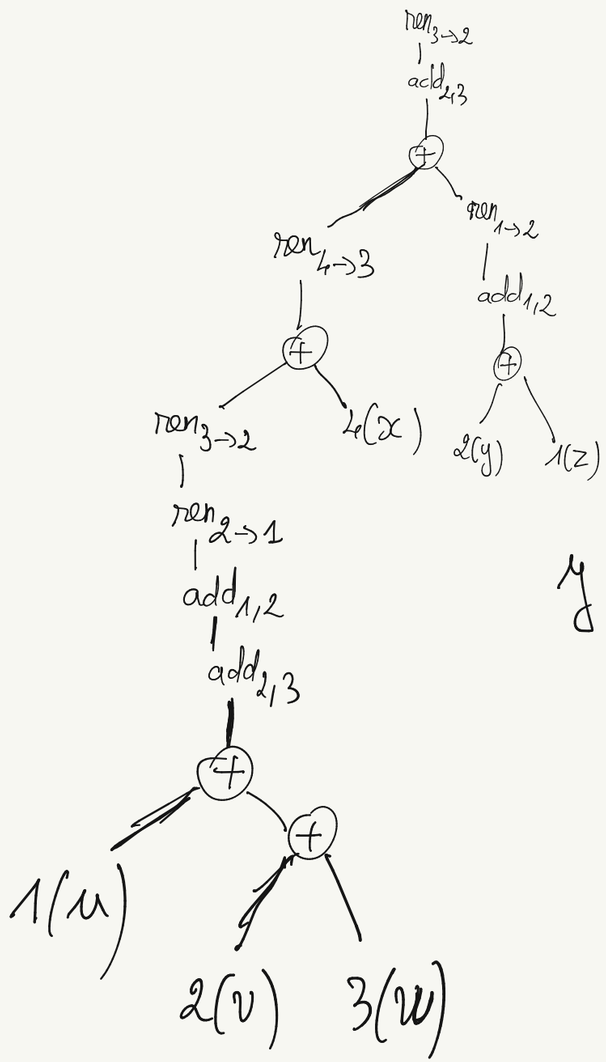


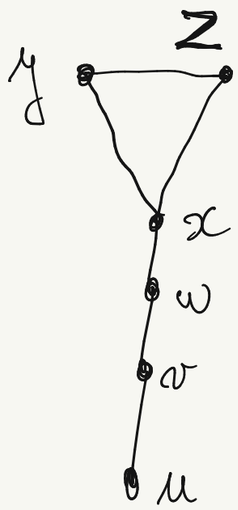
important: how the neighborhoods are



Layout

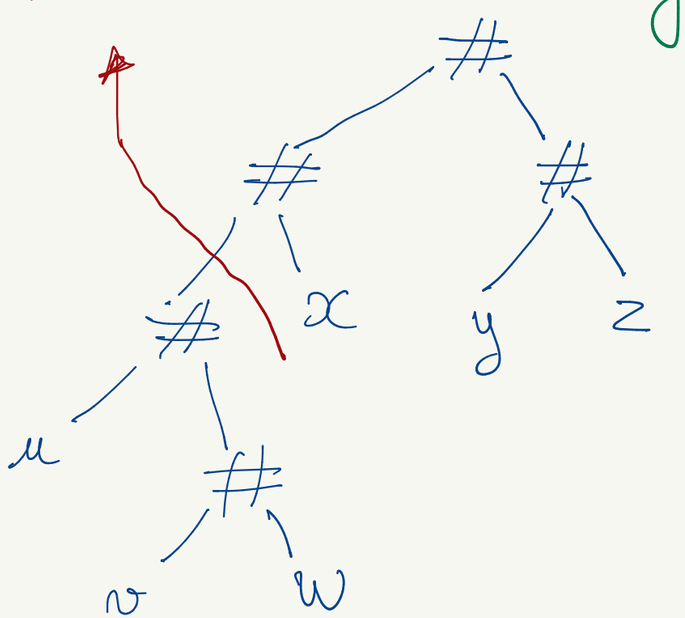
Look at Neighborhoods





important: how the neighborhoods are

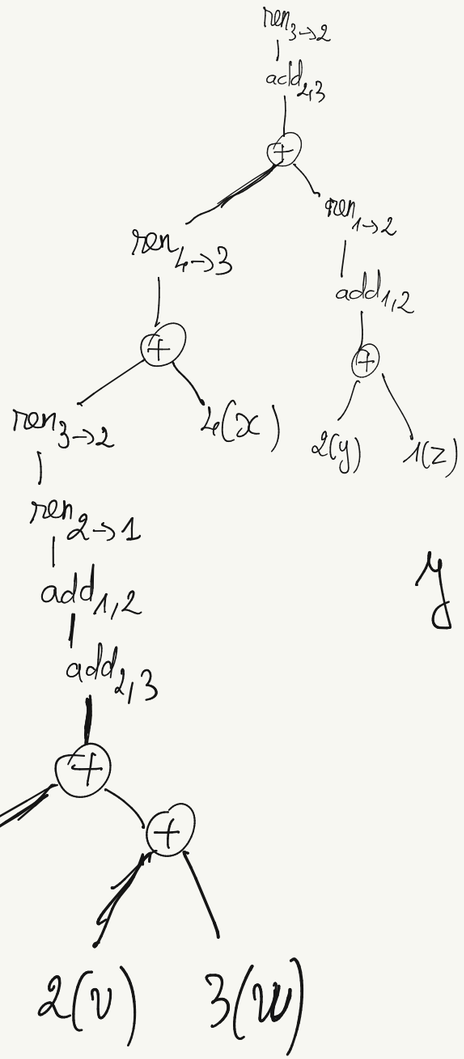
any function describing it is good

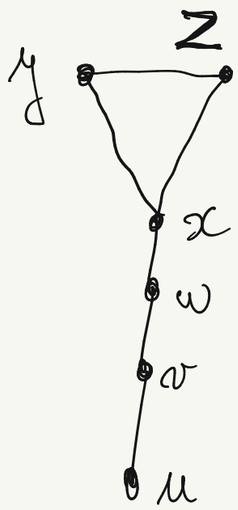


Layout

- cwd : number of twin classes (with a symmetry).

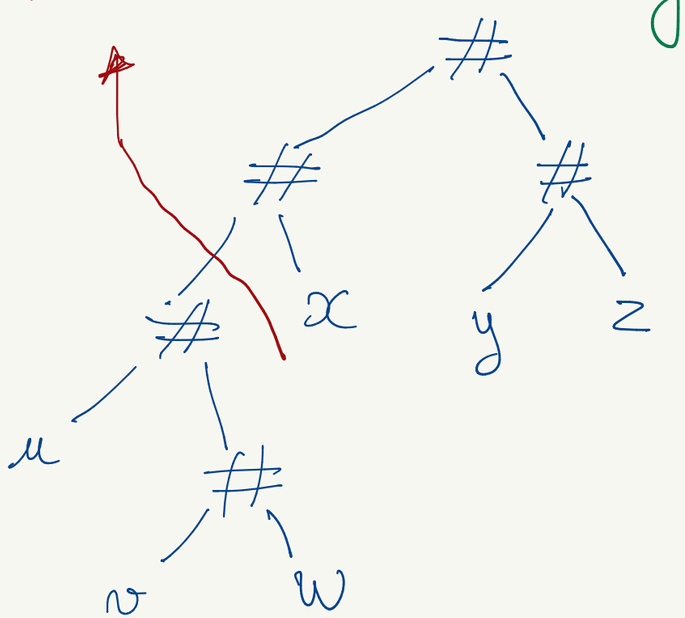
Look at Neighborhoods





important: how the neighborhoods are

any function describing it is good

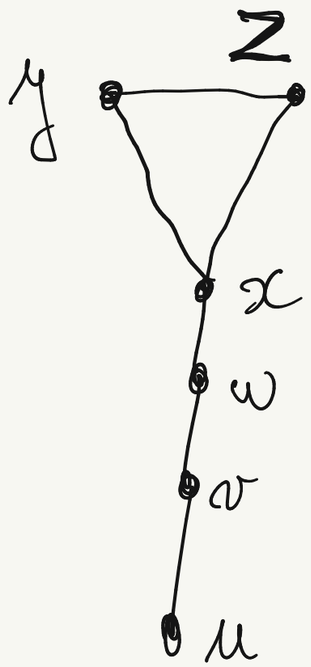


Layout

- cwd : number of twin classes (with a symmetry).
- others : distinct neighborhoods, rank-width, boolean-width, #-join, ...

5. Why Rank-width

RANK-WIDTH



$\forall e \in E(T)$:

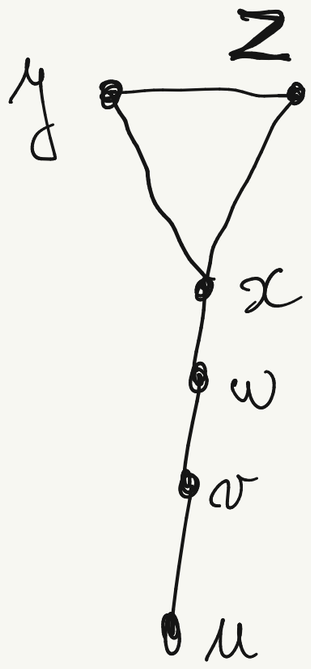
$A[X_e, \bar{X}_e]$

	x	y	z
u	0	0	0
v	0	0	0
w	1	0	0

$w(e) = rk\left(A[X_e, \bar{X}_e]\right)$

Layout : (T, L), $L : V(G) \longrightarrow$ leaves of T

RANK-WIDTH



$\forall e \in E(T)$:

$A[X_e, \bar{X}_e]$

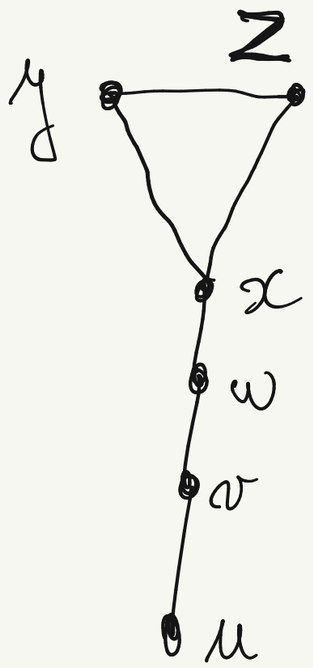
	x	y	z
u	0	0	0
v	0	0	0
w	1	0	0

$w(e) = rk\left(A[X_e, \bar{X}_e]\right)$

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

Layout : (T, L), $L : V(G) \longrightarrow$ leaves of T

RANK-WIDTH



$\forall e \in E(T)$:

$A[X_e, \bar{X}_e]$

	x	y	z
u	0	0	0
v	0	0	0
w	1	0	0

$w(e) = rk\left(A[X_e, \bar{X}_e]\right)$

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

$\bar{X}_e$

X_e

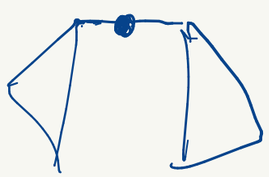
$$rwd(G) = \min_{(T,L)\ \text{layout}} wd(T,L)$$

Layout : (T, L), $L : V(G) \longrightarrow$ leaves of T

RANK-WIDTH

Proposition 5.1 $\quad rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$

Proof: - (T, L) a layout, root it at r = subdivide an edge.



- For $u \in V(T)$, compute t_u for G_u s.t. $lab(x) = lab(y) \iff N(x) \cap \overline{V(G_u)} = N(y) \cap \overline{V(G_u)}$

RANK-WIDTH

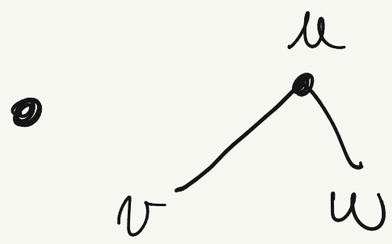
Proposition 5.1 $\quad rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$

Proof: - (T, L) a layout, root it at r.

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

$lab(x) = lab(y) \iff N(x) \cap \overline{V(G_u)} = N(y) \cap \overline{V(G_u)}$

- at most $2^{rwd(G)}$ such twin classes.



$\bullet$: $t'_u = \bigodot_{(i,j') \in R_u} add_{i,j'} \left(t_w \oplus \bigodot_{i \neq j} ren_{i \to j'}(t_w) \right)$

$R_u := \{(i,j') : \exists x \in G_v, y \in G_w, xy \in E,$ labeled resp. i and $j\}$ $\square$

RANK-WIDTH

Proposition 5.1 $\quad rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$

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

Proof: same proof idea as for clique-width. □

RANK-WIDTH

Proposition 5.1 $\quad rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$

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

Proposition 5.3 : $\quad rwd(G) = \max \{ rwd(H) \}$

$H \subseteq_i G$.

prime wrt split-decomposition

RANK-WIDTH

Proposition 5.1 $\quad rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$

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

Proposition 5.3 : $\quad rwd(G) = \max \{ rwd(H) \}$

$H \subseteq_i G$.

prime wrt split-decomposition

Observation 5.1 $\quad H \subseteq_i G$, then $rwd(H) \leq rwd(G)$.

Proposition 5.4 $\quad rwd(G) \leq tw(G) + 1$.

RANK-WIDTH

Proposition 5.1 $\quad rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$

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

Proposition 5.3 : $rwd(G) = \max \{ rwd(H) \}$
$H \subseteq_i G$.
prime wrt split-decomposition

Observation 5.4: $H \subseteq_i G$, then $rwd(H) \leq rwd(G)$.

But more operations.

RANK-WIDTH and STRUCTURE

- $X \begin{bmatrix} & \\ & \end{bmatrix}$

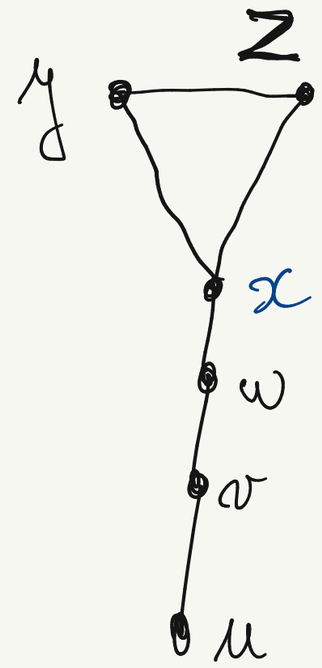
- adding a row to other rows does not increase rank.
- similarly for column.

RANK-WIDTH and STRUCTURE

- 

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

	x	y	z	w	v	u
x	0	1	1	1	0	0
y	1	0	1	0	0	0
z	1	1	0	0	0	0
w	1	0	0	0	1	0
v	0	0	0	1	0	0
u	0	0	0	0	1	0

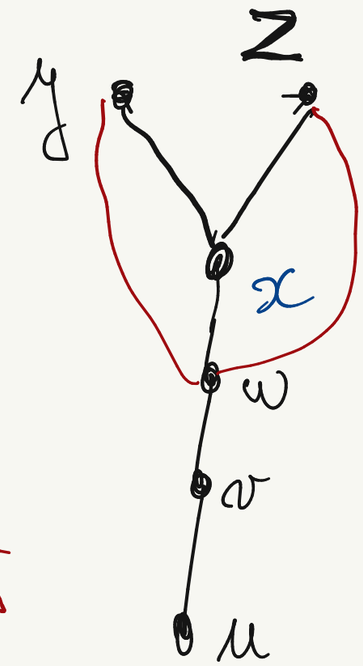


RANK-WIDTH and STRUCTURE

- 

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

	x	y	z	w	v	u
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
v	0	0	0	1	0	0
u	0	0	0	0	1	0



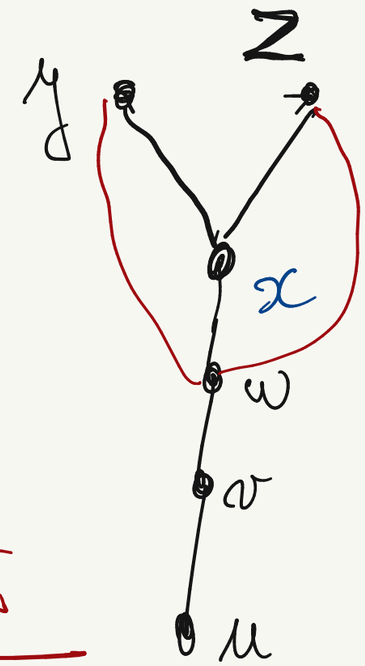
$G * x$ = local complementation at x

RANK-WIDTH and STRUCTURE

- $X \begin{bmatrix} & \\ & \end{bmatrix}$

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

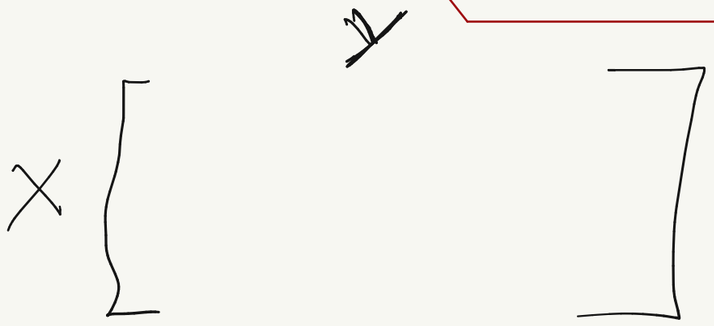
	x	y	z	w	v	u
x	0	1	1	1	0	0
$x+y$	1	0	0	1	0	0
$x+z$	1	0	0	1	0	0
$x+w$	1	1	1	0	1	0
v	0	0	0	1	0	0
u	0	0	0	0	1	0



$G * x$ = local complementation at x

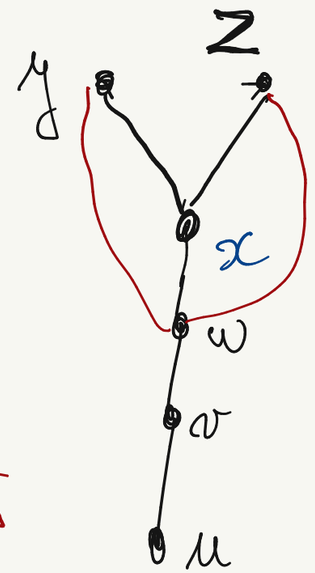
$$G * x = (G \setminus N(x)) \cup \overline{G[N(x)]}$$

RANK-WIDTH and STRUCTURE



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

	x	y	z	w	v	u
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
v	0	0	0	1	0	0
u	0	0	0	0	1	0



local complementation at x

$$G * x = (G \setminus N(x)) \cup \overline{G[N(x)]}$$

- H vertex-minor of G if $H \subseteq_i G'$,
  G' = successive local complementations on G.

RANK-WIDTH and STRUCTURE

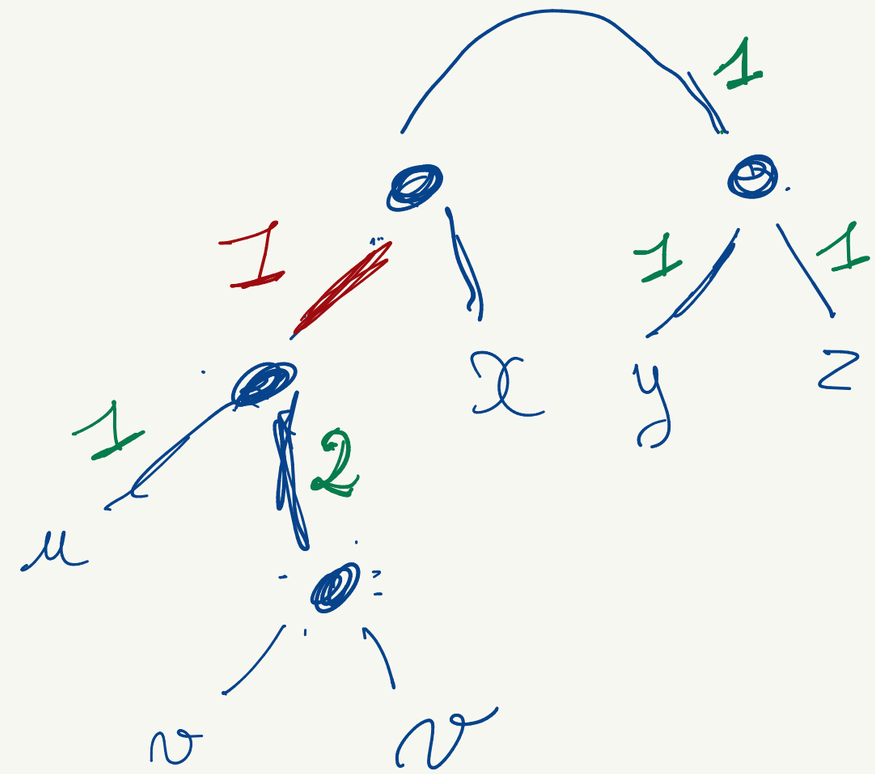
Proposition 5.5: H ~~vertex~~-minor, then $rwd(H) \leq rwd(G)$.

Proof sketch.

	x	y	z
u	0	0	0
v	0	0	0
w	1	0	0

$G \ast x$

	x	y	z	x	x
		+		+	
u	0	0	0		
v	0	0	0		
w	1	1	1		



RANK-WIDTH and STRUCTURE

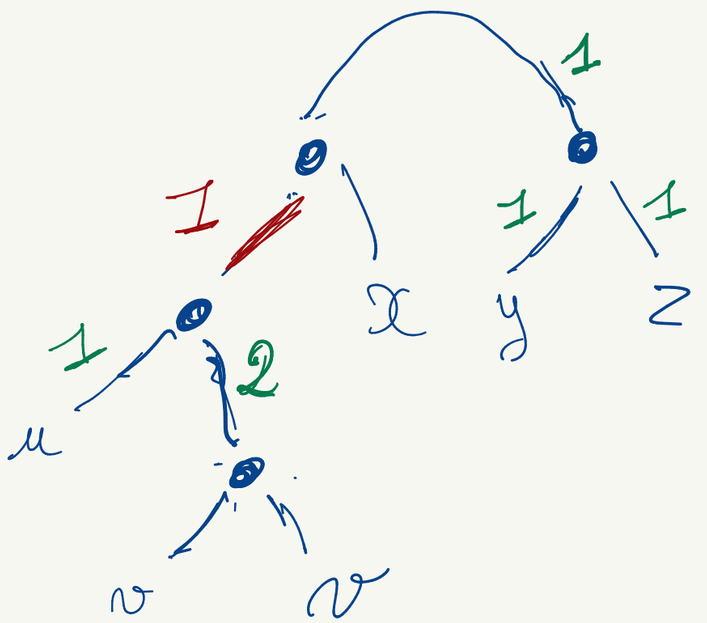
Proposition 5.5 : H ~~vertex~~-minor, then $rwd(H) \leq rwd(G)$.

Proof sketch.

	x	y	z
u	0	0	0
v	0	0	0
w	1	0	0

$G * x$

	x	y	z
u	0	0	0
v	0	0	0
w	1	1	1



Old notion : already used by Bouchet to characterize circle graphs by a finite list of obstructions.

RANK-WIDTH and STRUCTURE

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

RANK-WIDTH and STRUCTURE

- 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

RANK-WIDTH and STRUCTURE

- 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:

RANK-WIDTH and STRUCTURE

- 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.

RANK-WIDTH and STRUCTURE

- 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

RANK-WIDTH and STRUCTURE

- 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
- The rank function is submodular and symmetric
  $\Rightarrow$ obstruction set system like brambles $\simeq$ tangles

RANK-WIDTH and STRUCTURE

- 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
- The rank function is submodular and symmetric
  - $\Rightarrow$ obstruction set system like brambles.
- HUM: Looks like a lot tree-width, but suited for dense graphs.

RANK-WIDTH and STRUCTURE

- 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
- The rank function is submodular and symmetric
  - $\Rightarrow$ obstruction set system like brambles.
- HUM: Looks like a lot tree-width, but suited for dense graphs.

YES INDEED GENERALISE PRELIMINARY RESULTS FROM GRAPH/MATROID MINOR THEORY

MERCI