Tree-layout based graph classes: the case of proper chordal graphs

Evangelos Protopapas

LIRMM, Université de Montpellier, CNRS, Montpellier, France

GROW 2022

Joint work with:

Christophe Paul

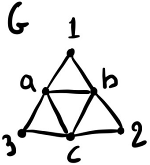
LIRMM, Université de Montpellier, CNRS, Montpellier, France

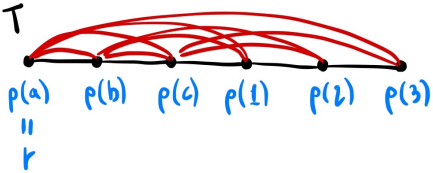
Tree-layouts

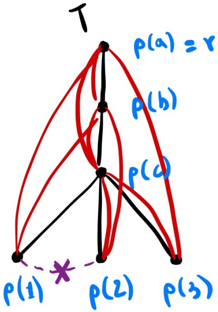
Tree-layout $\mathbf{T} = (T, r, \rho)$ of a graph G:

Tree-layouts

Tree-layout $\mathbf{T} = (T, r, \rho)$ of a graph G:

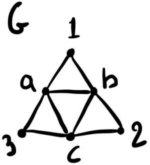


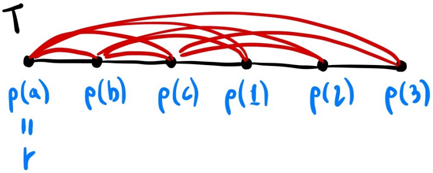


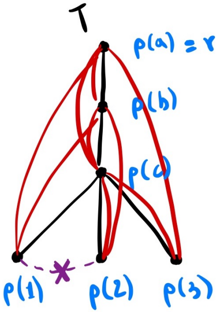


Tree-layouts

Tree-layout $\mathbf{T} = (T, r, \rho)$ of a graph G:







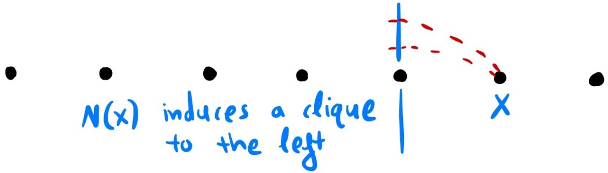
$u <_{\mathbf{T}} v$ iff $\rho(x)$ is an ancestor of $\rho(y)$ in T.
$\mathbf{T}$ is a **tree-layout** of G iff $\forall xy \in E(G)$: $x <_{\mathbf{T}} y$ or $y <_{\mathbf{T}} x$.

Tree-layout characterization of chordal graphs

G **chordal:** Admits a **simplicial elimination ordering**.

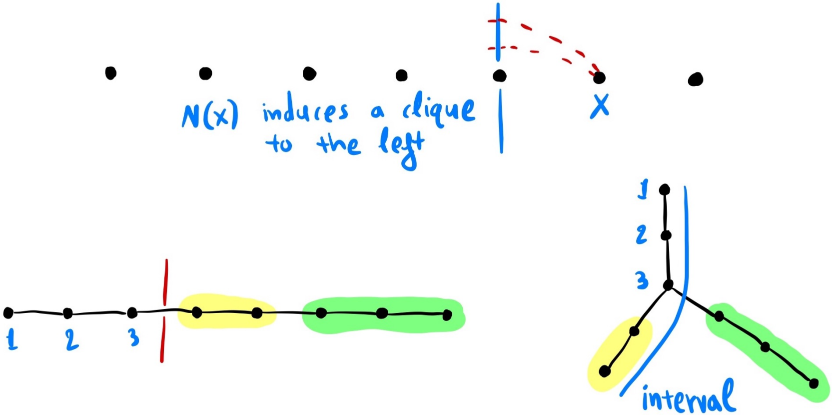
Tree-layout characterization of chordal graphs

G **chordal:** Admits a **simplicial elimination ordering**.



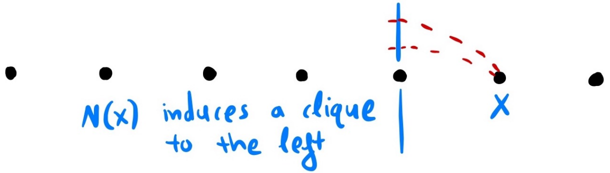
Tree-layout characterization of chordal graphs

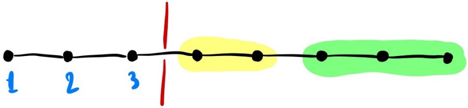
G **chordal:** Admits a **simplicial elimination ordering**.

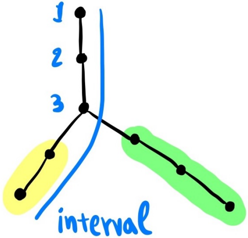


Tree-layout characterization of chordal graphs

G **chordal:** Admits a **simplicial elimination ordering**.







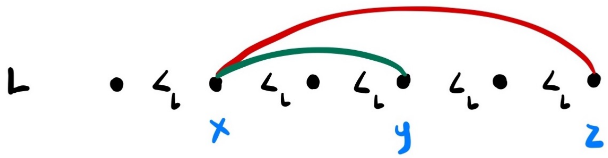
G **chordal:** Admits tree-layout where the graph induced by **every** root-leaf path is an **interval** graph.

Layout based classes

Specific **configuration** of layouts $\Rightarrow$ Graph class **characterization**.

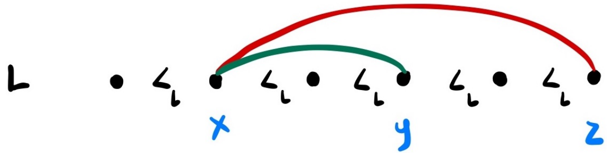
Layout based classes

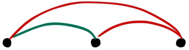
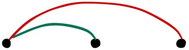
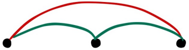
Specific **configuration** of layouts $\Rightarrow$ Graph class **characterization**.



Layout based classes

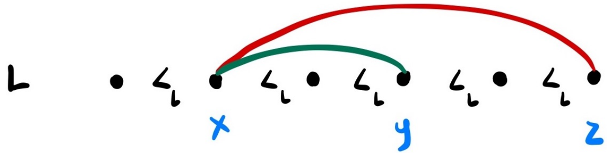
Specific **configuration** of layouts ⇒ Graph class **characterization**.

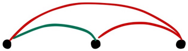
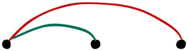
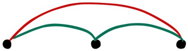


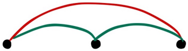
Configuration	**Layouts**	**Tree-Layouts**
	Chordal	Chordal
	Interval	**Chordal**
	Proper Interval	???

Layout based classes

Specific **configuration** of layouts $\Rightarrow$ Graph class **characterization**.



Configuration	**Layouts**	**Tree-Layouts**
	Chordal	Chordal
	Interval	**Chordal**
	Proper Interval	???

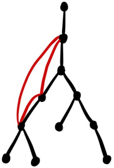
: **indifference property**

Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.

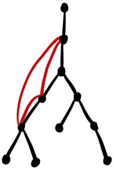
Proper Chordal graphs

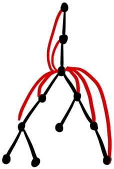
G is a **proper chordal** iff it admits an **indifference tree-layout**.



Proper Chordal graphs

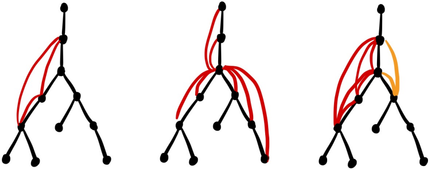
G is a **proper chordal** iff it admits an **indifference tree-layout**.





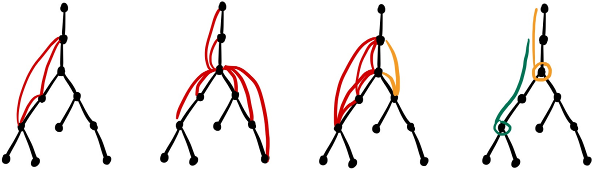
Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.



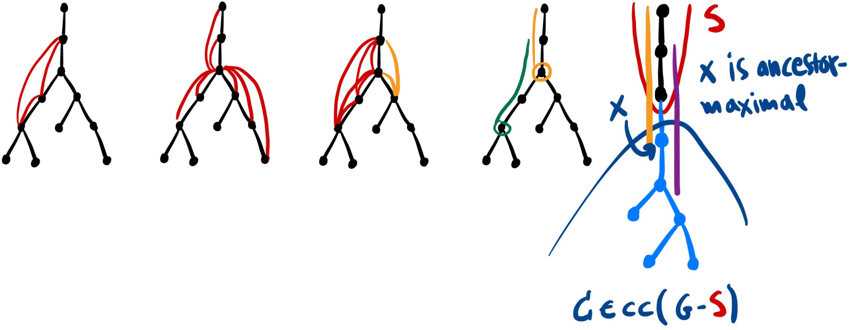
Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.



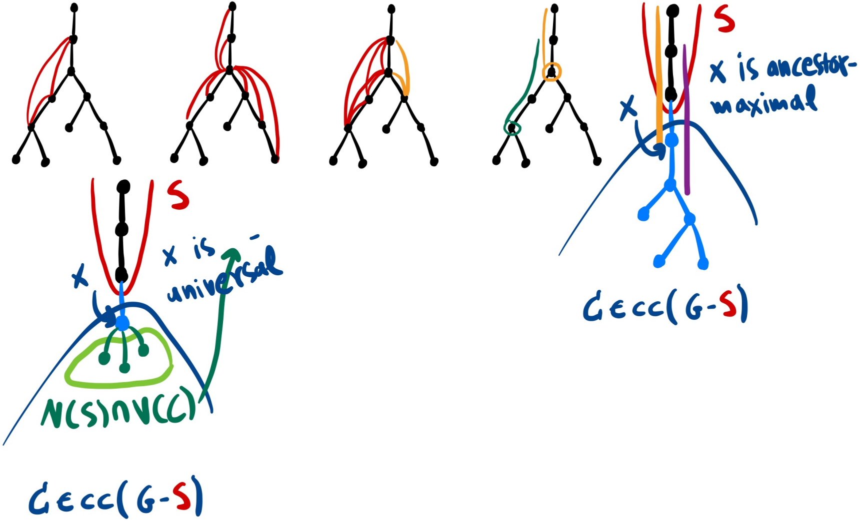
Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.



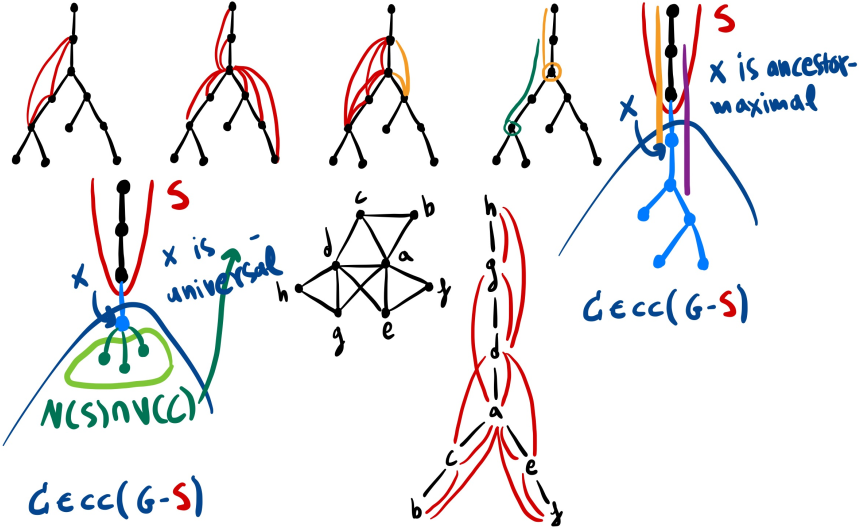
Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.



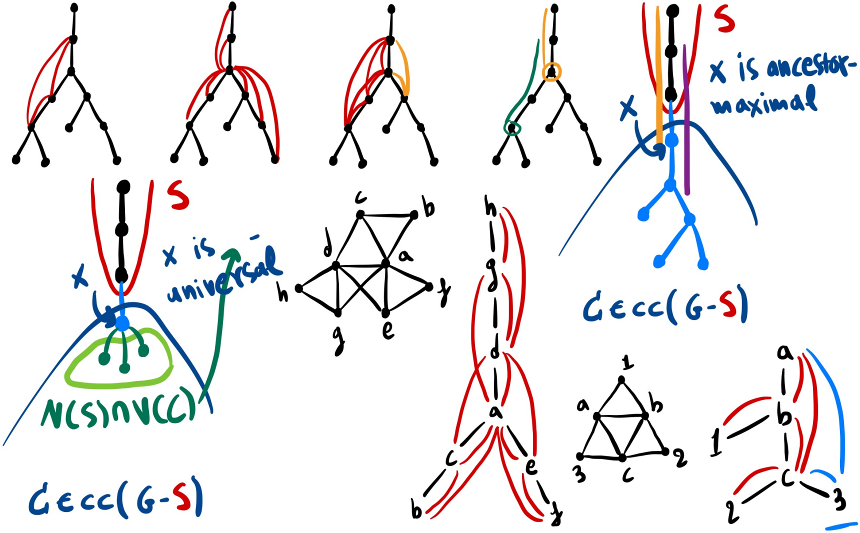
Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.



Proper Chordal graphs

G is a **proper chordal** iff it admits an **indifference tree-layout**.



Positioning of Proper Chordal graphs

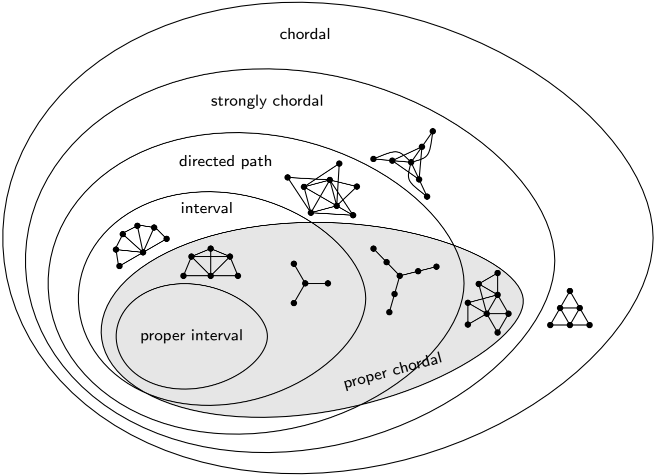
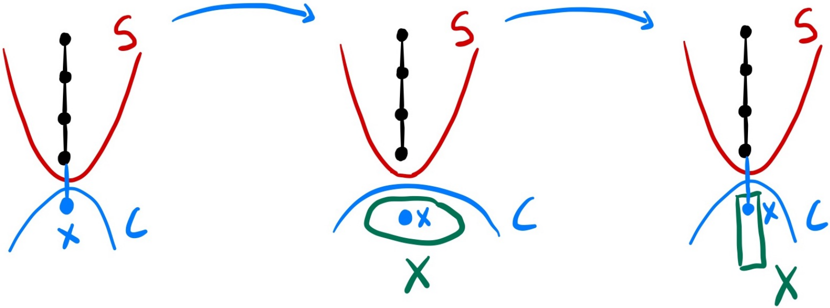
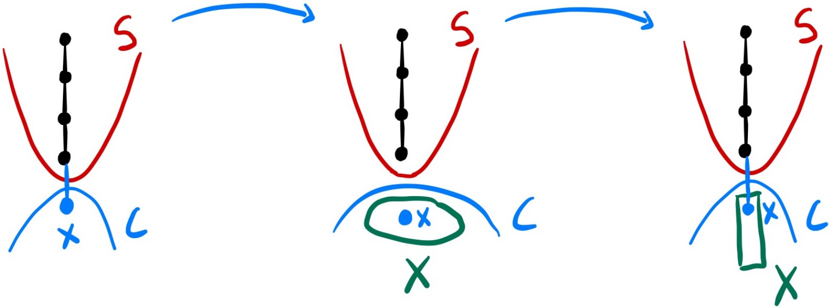


Figure: Relationship between proper chordal graphs and subclasses of chordal graphs.

Blocks

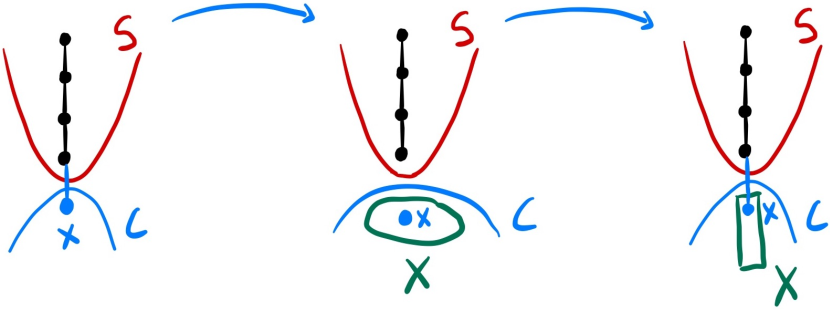


Blocks



- S is the set of ancestors of x on some indiff. tl. $\mathbf{T}$; $C \in \mathrm{cc}(G - S)$;
- X is the set of ancestor-maximal and $(N(S) \cap V(C))$-universal vertices in C.

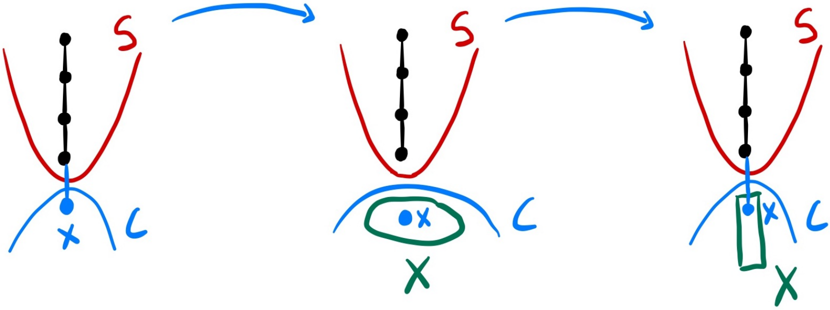
Blocks



- S is the set of ancestors of x on some indiff. tl. $\mathbf{T}$; $C \in \mathrm{cc}(G - S)$;
- X is the set of ancestor-maximal and $(N(S) \cap V(C))$-universal vertices in C.

We can prove that X has to appear **first** and **consecutively**.

Blocks

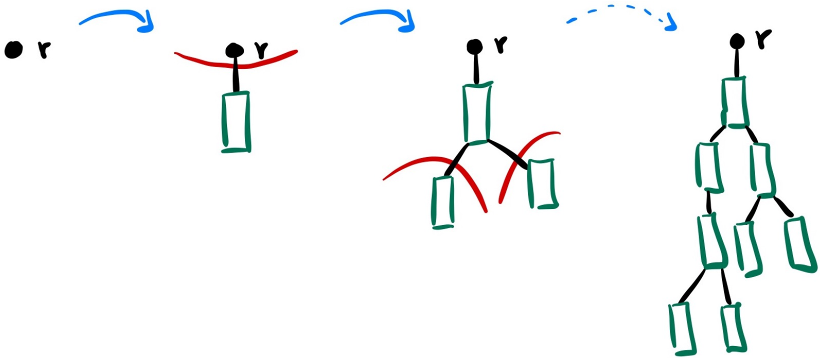


- S is the set of ancestors of x on some indiff. tl. $\mathbf{T}$; $C \in \mathrm{cc}(G - S)$;
- X is the set of ancestor-maximal and $(N(S) \cap V(C))$-universal vertices in C.

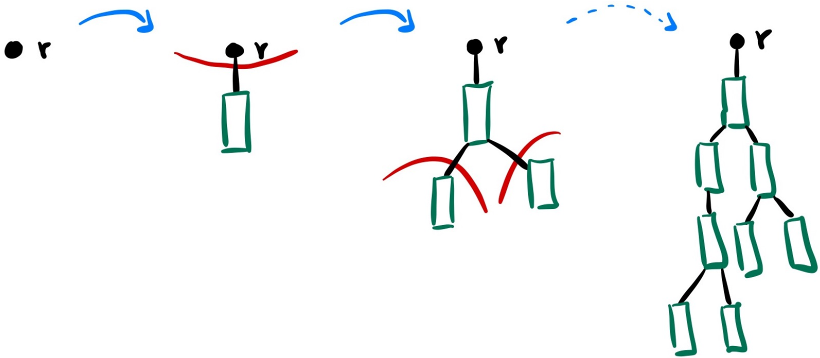
We can prove that X has to appear **first** and **consecutively**.

We call X a **block**.

Block Tree

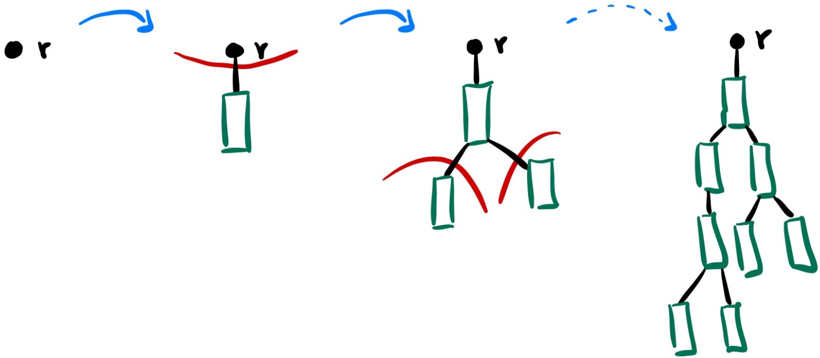


Block Tree



There is a **unique** block tree rooted at each $r \in V(G)$.

Block Tree

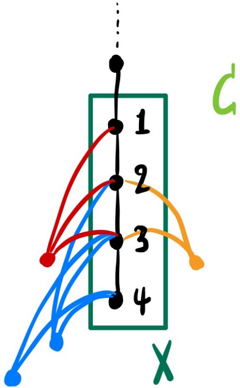


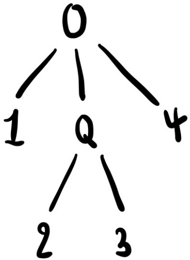
There is a **unique** block tree rooted at each $r \in V(G)$.

To obtain an indifference tree-layout, it remains to:

1. **Order** vertices within blocks;
2. **Properly attach** children blocks to parent block.

Ordering blocks





1 2 3

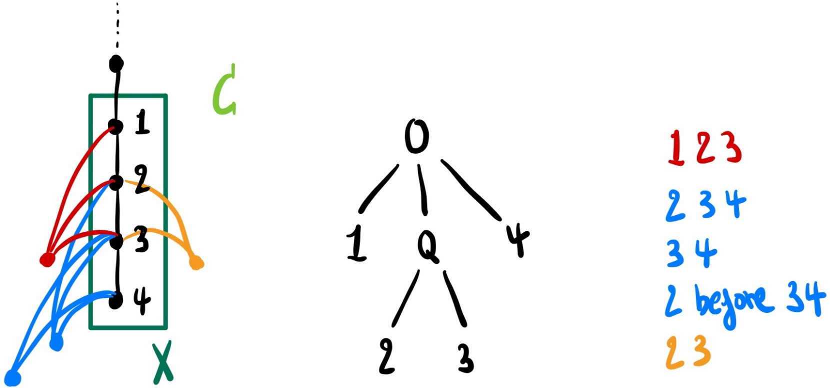
2 3 4

3 4

2 before 34

2 3

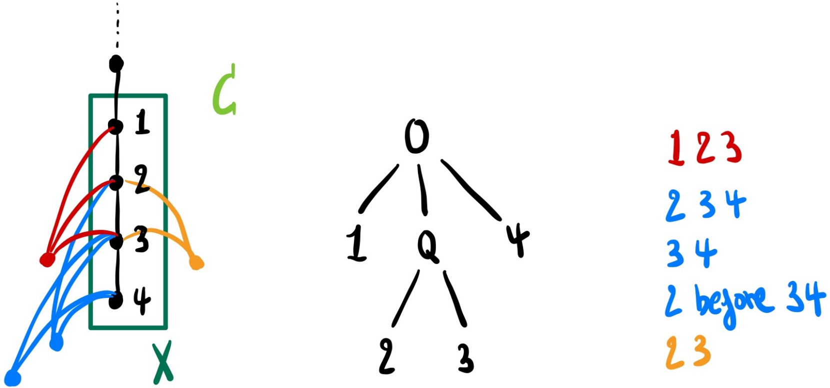
Ordering blocks



The **order** of X needs to satisfy certain **convexity** conditions:

- $N(u) \cap X$, where $u \in V(C - X)$ has to appear consecutively;
- For each component of $C - X$ we need to respect the **inclusion ordering** (maximal to minimal neighbourhood).

Ordering blocks

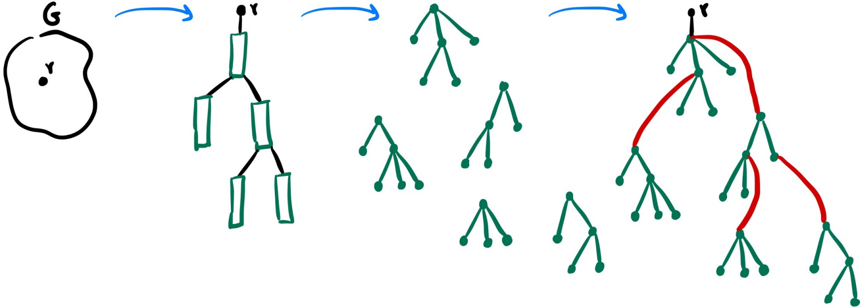


The **order** of X needs to satisfy certain **convexity** conditions:

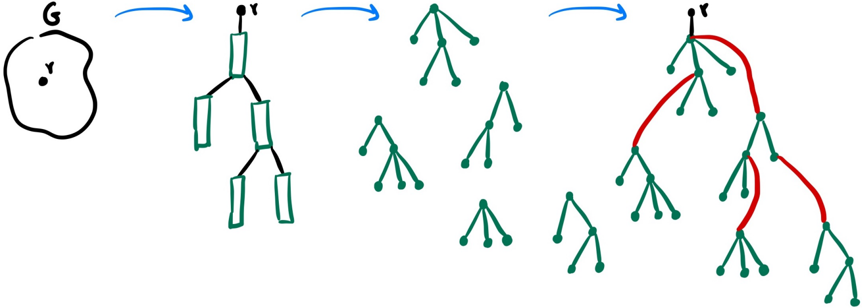
- $N(u) \cap X$, where $u \in V(C - X)$ has to appear consecutively;
- For each component of $C - X$ we need to respect the **inclusion ordering** (maximal to minimal neighbourhood).

Unique OPQ-tree represents all **possible permutations** of X.

Canonical representation

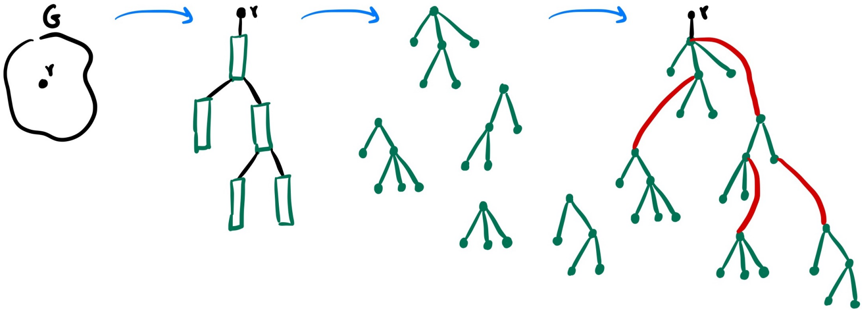


Canonical representation



Unique OPQ-hierarchy represents all **indifference tree-layouts** of G **rooted** at $r \in V(G)$.

Canonical representation



Unique OPQ-hierarchy represents all **indifference tree-layouts** of G **rooted** at $r \in V(G)$.

Remark: We can compute it in **poly-time** and has size **linear** in $|G|$.

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition for a graph G:

- For every $u \in V(G)$;
- Compute **block tree** rooted at u;
- Compute OPQ-tree corresponding to each **block**;
- Verify that each block has a valid **convex order**.

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition for a graph G:

- For every $u \in V(G)$;
- Compute **block tree** rooted at u;
- Compute OPQ-tree corresponding to each **block**;
- Verify that each block has a valid **convex order**.

What about GRAPH ISOMORPHISM?

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition for a graph G:

- For every $u \in V(G)$;
- Compute **block tree** rooted at u;
- Compute OPQ-tree corresponding to each **block**;
- Verify that each block has a valid **convex order**.

What about Graph Isomorphism?

We can solve it...

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition for a graph G:

- For every $u \in V(G)$;
- Compute **block tree** rooted at u;
- Compute OPQ-tree corresponding to each **block**;
- Verify that each block has a valid **convex order**.

What about GRAPH ISOMORPHISM?

We can solve it... maybe...

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition for a graph G:

- For every $u \in V(G)$;
- Compute **block tree** rooted at u;
- Compute OPQ-tree corresponding to each **block**;
- Verify that each block has a valid **convex order**.

What about Graph Isomorphism?

We can solve it... maybe...

Depends on whether we can do **isomorphism** for OPQ-hierarchies.

Recognition & Isomorphism

Recognition of proper chordal graphs is in P.

Corollary of the **canonical representation**!

Recognition for a graph G:

- For every $u \in V(G)$;
- Compute **block tree** rooted at u;
- Compute OPQ-tree corresponding to each **block**;
- Verify that each block has a valid **convex order**.

What about GRAPH ISOMORPHISM?

We can solve it... maybe...

Depends on whether we can do **isomorphism** for OPQ-hierarchies.

Lemma

Two proper chordal graphs G and H are isomorphic iff there exist $u \in V(G)$ and $v \in V(H)$ such that the OPQ*-hierarchies rooted at u and v are isomorphic.*

Conclusion

Recap:

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Questions:

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Questions:

- Combinatorics of the **root** of an indifference tree-layout?

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Questions:

- Combinatorics of the **root** of an indifference tree-layout?
- What about other problems on Proper Chordal graphs?

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Questions:

- Combinatorics of the **root** of an indifference tree-layout?
- What about other problems on Proper Chordal graphs?
- Hamiltonian Cycle? We can solve it when the input graph is also a Split graph, but we failed in general.

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Questions:

- Combinatorics of the **root** of an indifference tree-layout?
- What about other problems on Proper Chordal graphs?
- HAMILTONIAN CYCLE? We can solve it when the input graph is also a Split graph, but we failed in general.
- Other classes of graphs? E.g. Tree Permutation graphs?

Conclusion

Recap:

- We introduced a new subclass of Chordal graphs, which we call **Proper Chordal graphs** via **tree-layouts**.
- We showed how to **canonically** represent all **indifference tree-layouts**.
- Results based on OPQ-**hierarchies**.

Questions:

- Combinatorics of the **root** of an indifference tree-layout?
- What about other problems on Proper Chordal graphs?
- HAMILTONIAN CYCLE? We can solve it when the input graph is also a Split graph, but we failed in general.
- Other classes of graphs? E.g. Tree Permutation graphs?
- Other applications of OPQ-hierarchies?

Thank you!