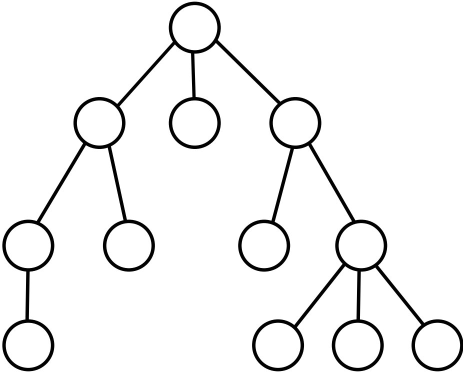
Computing Treedepth

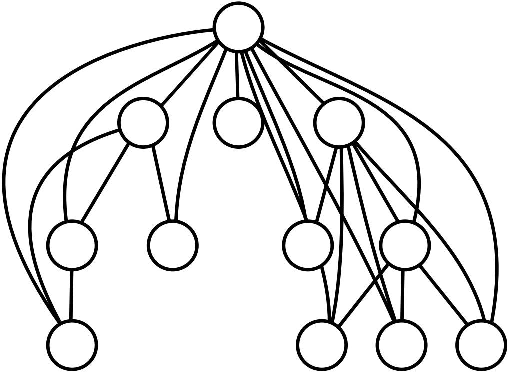
Felix Reidl, Peter Rossmanith, **Fernando Sánchez Villaamil**
Somnath Sikdar

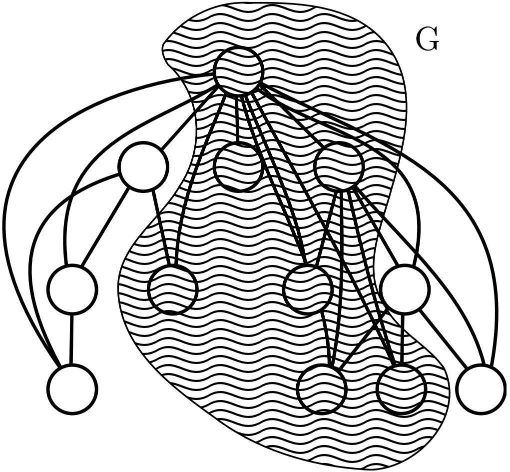
RWTH Aachen

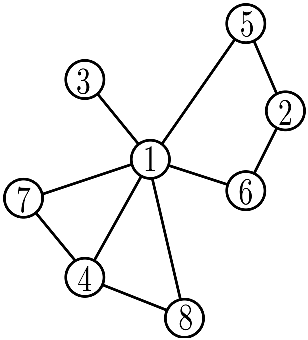
March 14, 2014

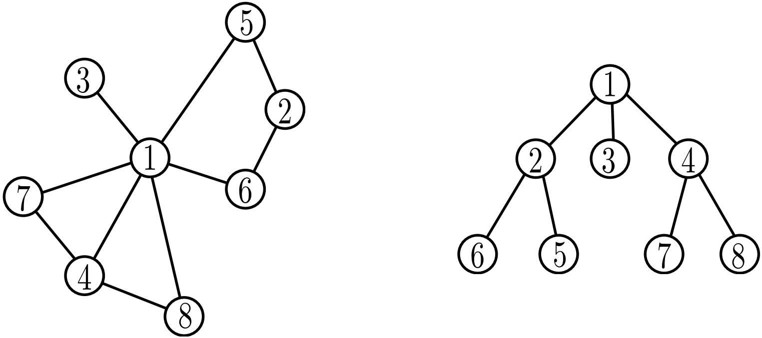
Treedepth is a width measure.

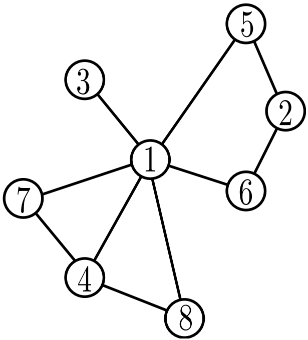


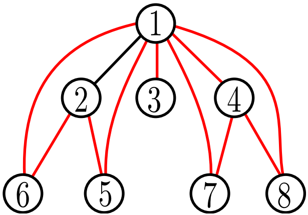


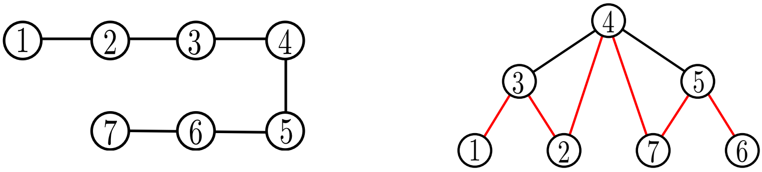

G


5
3
2
1
7
6
4
8


5
3
2
1
7
6
4
8
1
2
3
4
6
5
7
8

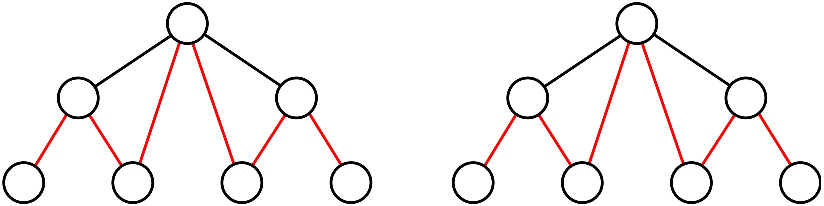

5
3
2
1
7
6
4
8


1
2
3
4
6
5
7
8

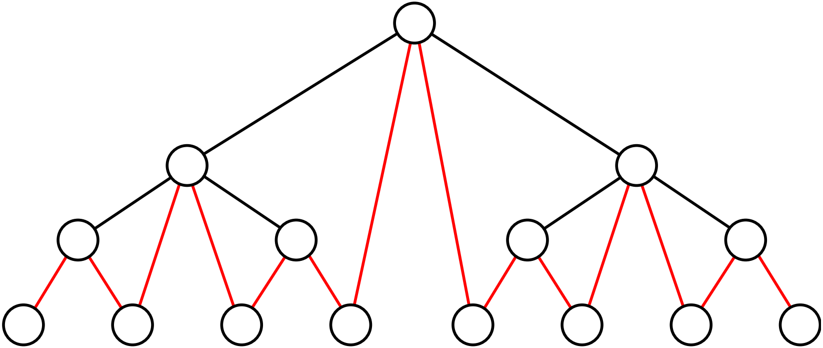


Treedepth $t \rightarrow$ Maximal path length $2^t - 1$.

Treedepth $t \rightarrow$ Maximal path length $2^t - 1$.



Treedepth $t \rightarrow$ Maximal path length $2^t - 1$.



Definition (Treedepth decomposition)

A *treedepth decomposition* of a graph G is a rooted forest F such that $V(G) \subseteq V(F)$ and $E(G) \subseteq E(clos(F))$.

Definition (Treedepth)

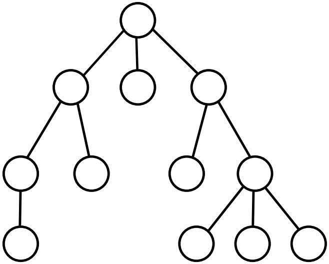
The *treedepth* $\mathbf{td}(G)$ of a graph G is the minimum height of any treedepth decomposition of G.

Equivalent notions

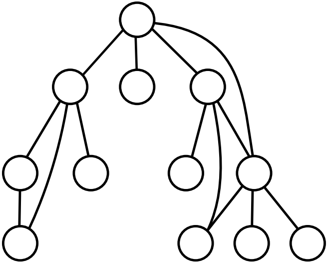
A graph G has treedepth at most t if and only if:

- G is a subgraph the closure of a tree (forest) of height t
- G is the subgraph of a trivially perfect graph with clique size at most t
- G has a centered coloring with t colors
- G has a ranked coloring with t colors

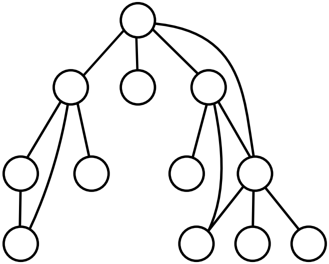
A DFS is a Treedepth decomposition



A DFS is a Treedepth decomposition

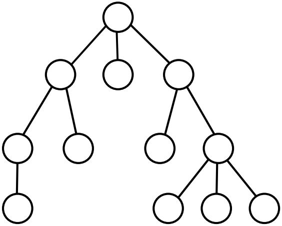


A DFS is a Treedepth decomposition

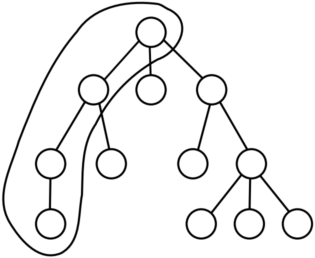


Treedepth $t \Rightarrow$ Maximal path length $2^t - 1 \Rightarrow 2^t$-approximation

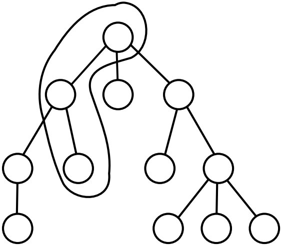
Treedepth to pathwidth



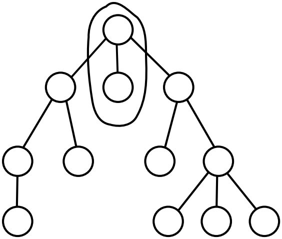
Treedepth to pathwidth



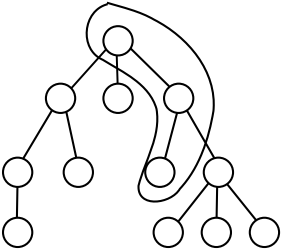
Treedepth to pathwidth



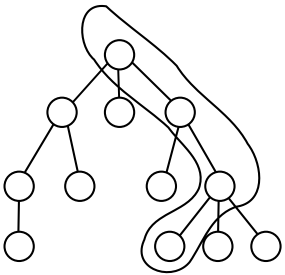
Treedepth to pathwidth



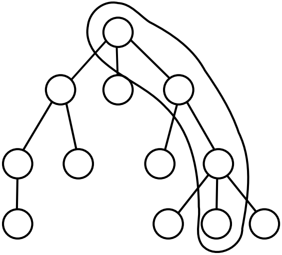
Treedepth to pathwidth



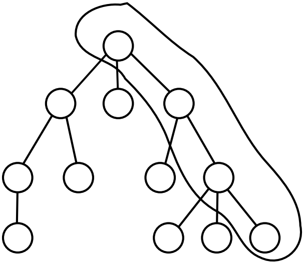
Treedepth to pathwidth



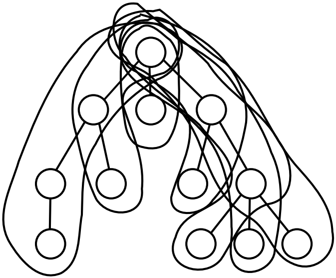
Treedepth to pathwidth



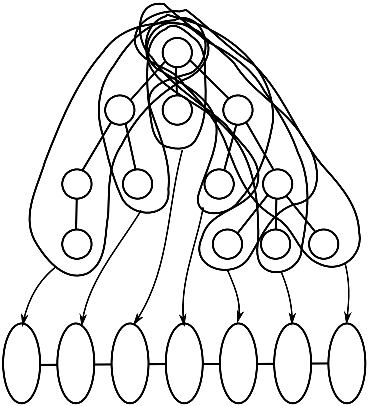
Treedepth to pathwidth



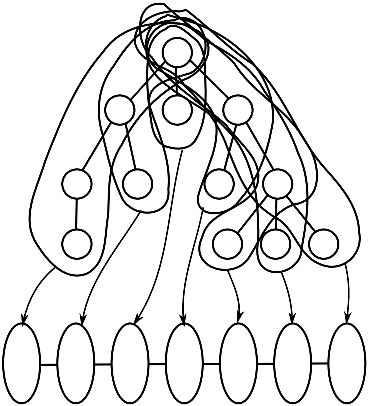
Treedepth to pathwidth



Treedepth to pathwidth



Treedepth to pathwidth



$$\mathbf{tw}(G) \leq \mathbf{pw}(G) \leq \mathbf{td}(G) - 1$$

Treedepth $t \Rightarrow$ Path decomposition of width $2^t - 2$

Known ways to compute the treedepth of a graph:

- In $f(t) \cdot n^3$ time by Robertson and Seymour.
- $\mathbf{tw}(G) \leq \mathbf{td}(G) - 1 \Rightarrow$ By Courcelle's Theorem $2^{2^{2^{\cdot^{\cdot^{\cdot^{t}}}}}} \cdot n$.
- Algorithm by Bodlaender et. al. with running time $2^{O(w^2 t)} \cdot n^2$.

Known ways to compute the treedepth of a graph:

- In $f(t) \cdot n^3$ time by Robertson and Seymour.
- $\mathbf{tw}(G) \leq \mathbf{td}(G) - 1 \Rightarrow$ By Courcelle's Theorem $2^{2^{2^{\cdot^{\cdot^{\cdot^{t}}}}}} \cdot n$.
- Algorithm by Bodlaender et. al. with running time $2^{O(w^2 t)} \cdot n^2$.

Problem

Is there a simple linear time algorithm to check $\mathbf{td}(G) \leq t$ *for fixed* t*?*

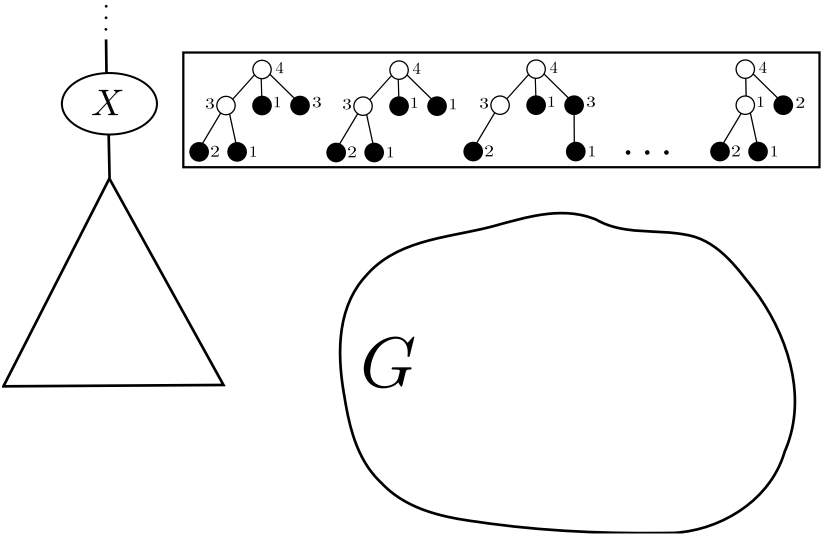
Our results:

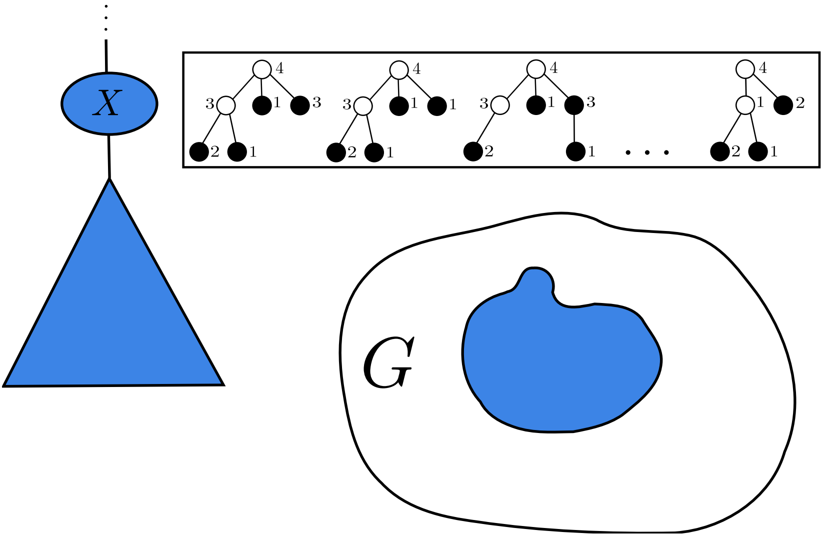
- A (relatively) simple direct algorithm in time $2^{2^{O(t)}} \cdot n$.
- A fast algorithm in time $2^{O(t^2)} \cdot n$.

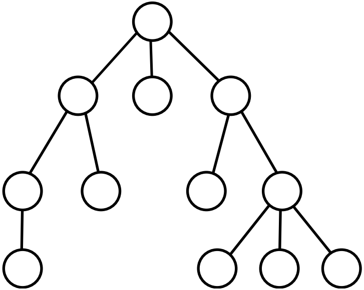
Our results:

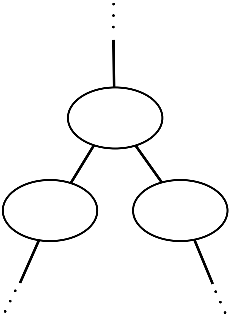
- A (relatively) simple direct algorithm in time $2^{2^{O(t)}} \cdot n$.
- A fast algorithm in time $2^{O(t^2)} \cdot n$.

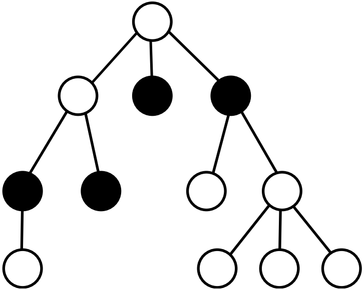
Both results follow from an algorithm on tree decompositions which runs in time $2^{O(wt)} \cdot n$.

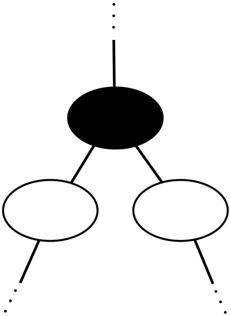

X
4
3
1
3
2
1
4
3
1
1
2
1
4
3
1
3
2
1
. . .
4
1
2
2
1
G

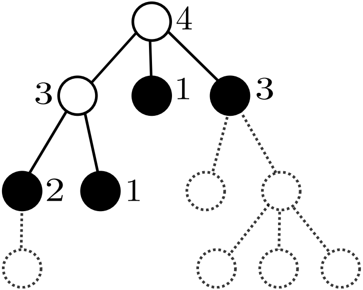

X
4
3
1
3
2
1
4
3
1
1
2
1
4
3
1
3
2
1
4
1
2
2
1
G

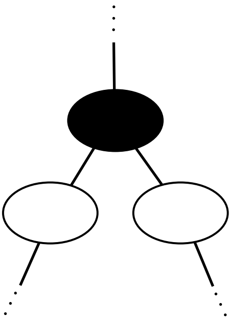


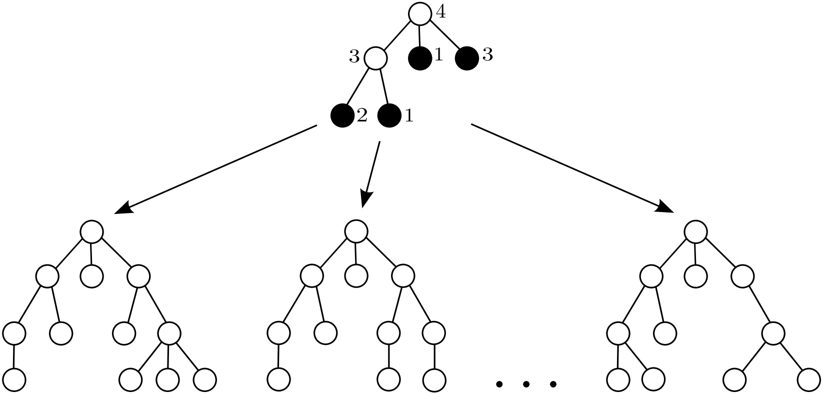


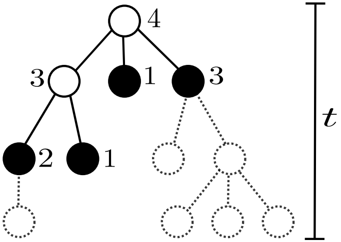


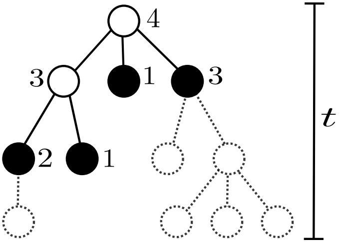



4
3
1
3
2
1

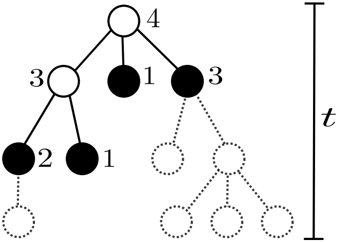



4
3
1
3
2
1
. . .


4
3
1
3
2
1
t



Any partial decomposition has less than *wt* nodes.

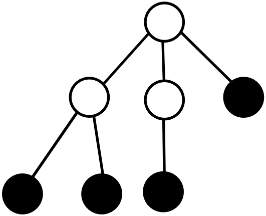


Any partial decomposition has less than *wt* nodes.

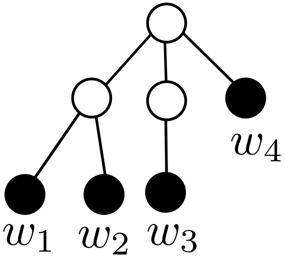
Counting

- How many trees with labeled leaves?
- How many height labelings per tree?

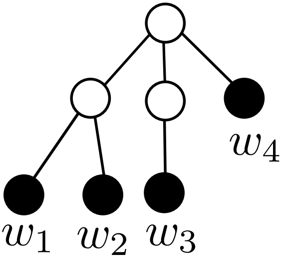
How many trees with labeled leaves?

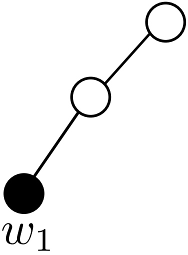


How many trees with labeled leaves?

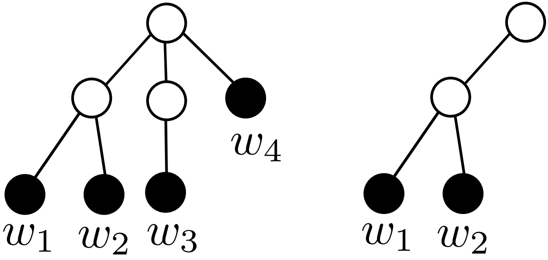


How many trees with labeled leaves?

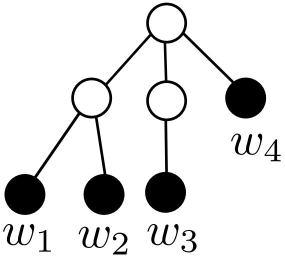




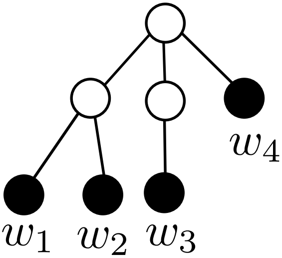
How many trees with labeled leaves?

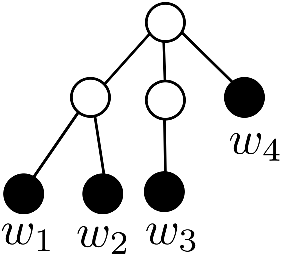


How many trees with labeled leaves?

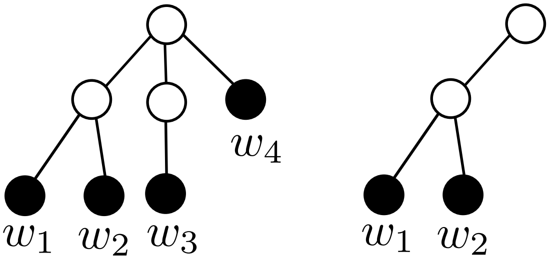


How many trees with labeled leaves?

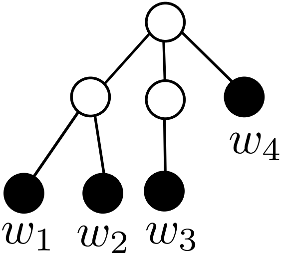


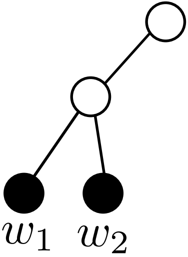


How many trees with labeled leaves?



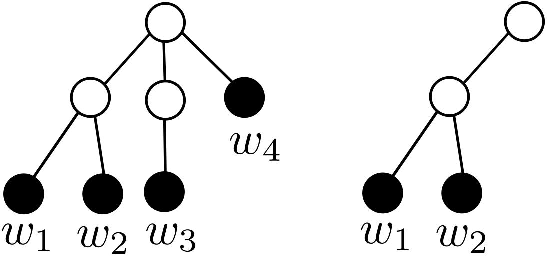
How many trees with labeled leaves?





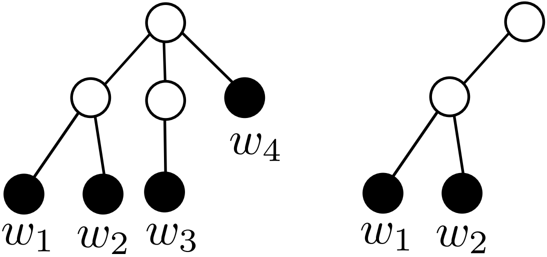
$$\prod_{i=1}^{w} it \cdot t$$

How many trees with labeled leaves?



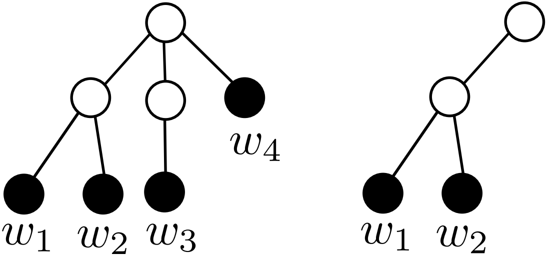
$$\prod_{i=1}^{w} it \cdot t \leq w! \cdot t^{2w}$$

How many trees with labeled leaves?



$$\prod_{i=1}^{w} it \cdot t \leq w! \cdot t^{2w} \leq 2^{2w \log t + w \log w}$$

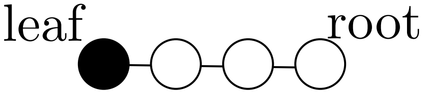
How many trees with labeled leaves?



$$\prod_{i=1}^{w} it \cdot t \leq w! \cdot t^{2w} \leq 2^{2w \log t + w \log w} = 2^{O(wt)}$$

How many height labelings per tree?

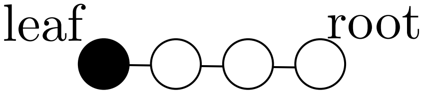
How many labeling can a path have?



$$(1, 2, 3, 4, 5, 6, \ldots, t)$$

How many height labelings per tree?

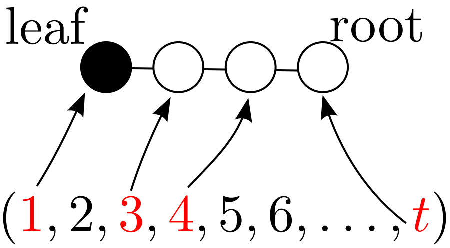
How many labeling can a path have?



$$(1, 2, 3, 4, 5, 6, \ldots, t)$$

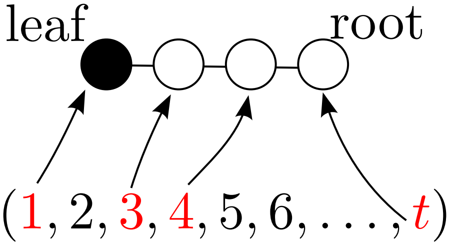
How many height labelings per tree?

How many labeling can a path have?



How many height labelings per tree?

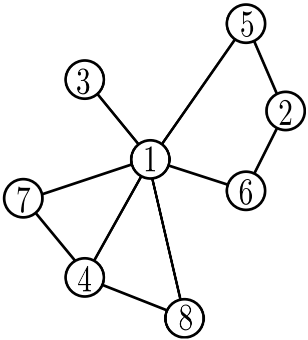
How many labeling can a path have?

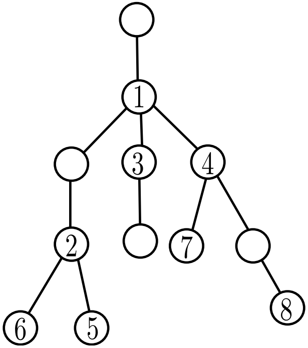


$$(2^t)^w = 2^{O(wt)}$$

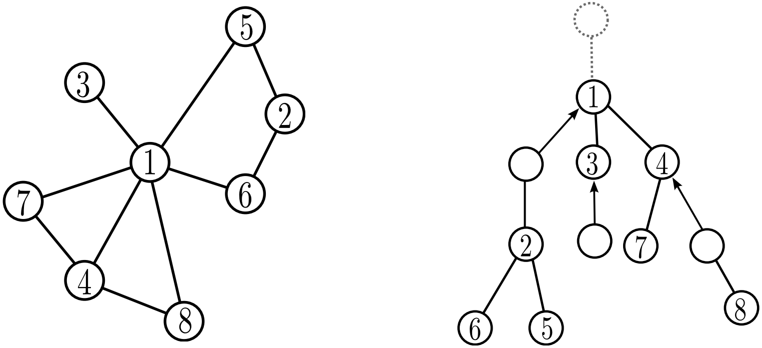
The total number of labeled trees is $2^{O(wt)} \cdot 2^{O(wt)} = 2^{O(wt)}$.

Non-trivially improvable treedepth decompositions

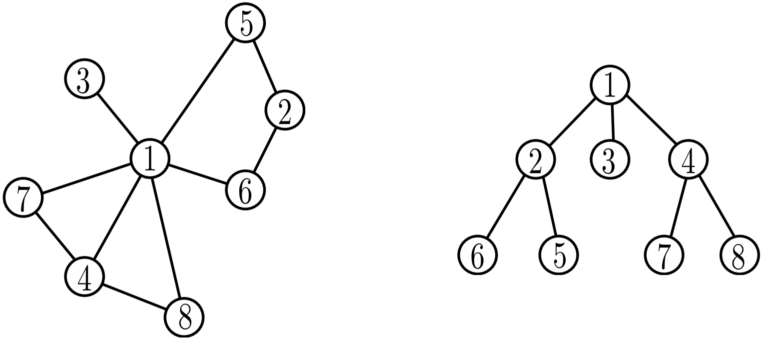


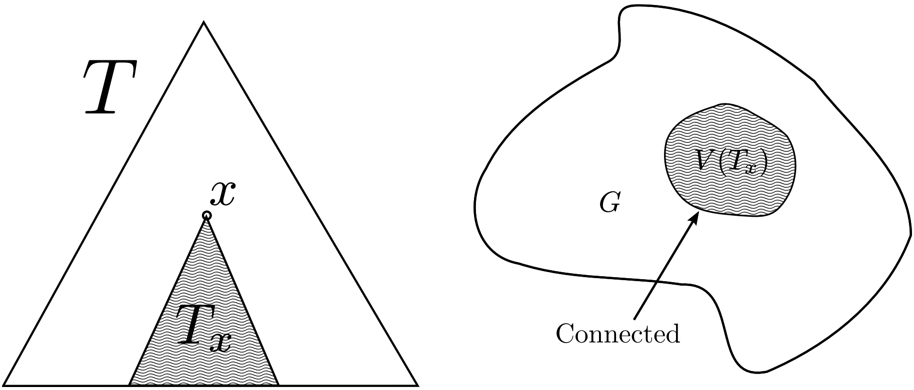


Non-trivially improvable treedepth decompositions

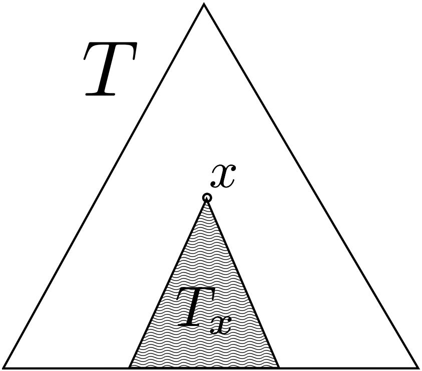


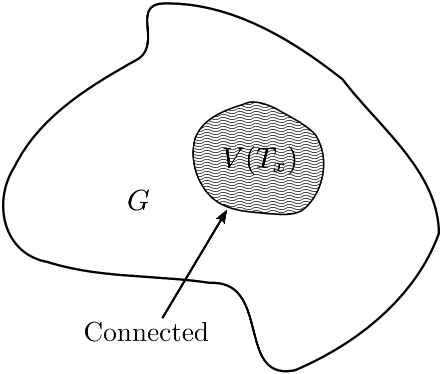
Non-trivially improvable treedepth decompositions





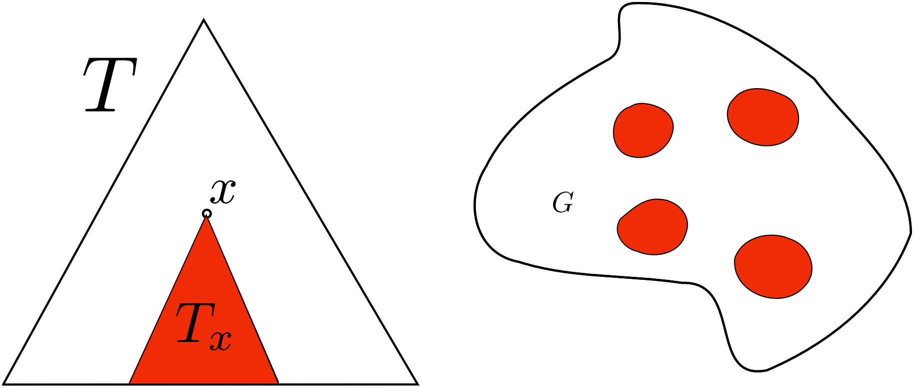
T
x
T_x
G
$V(T_x)$
Connected

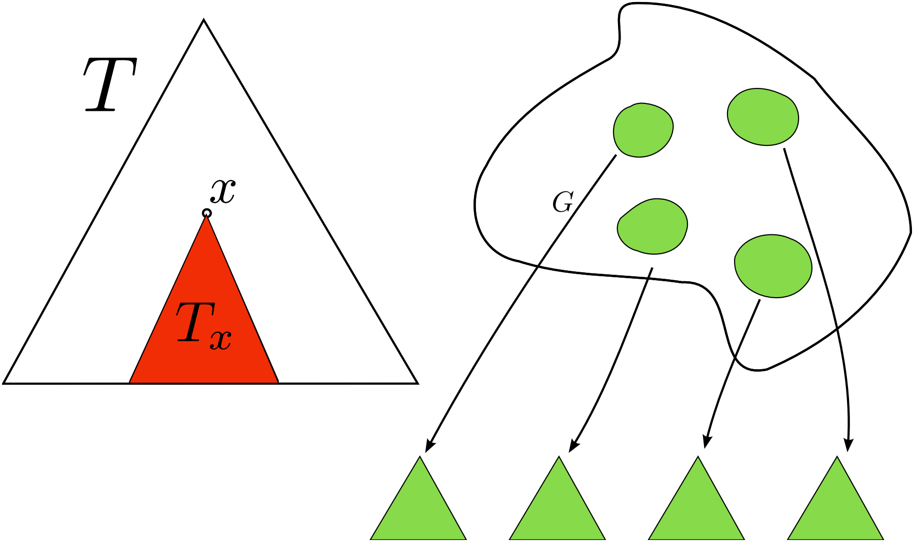




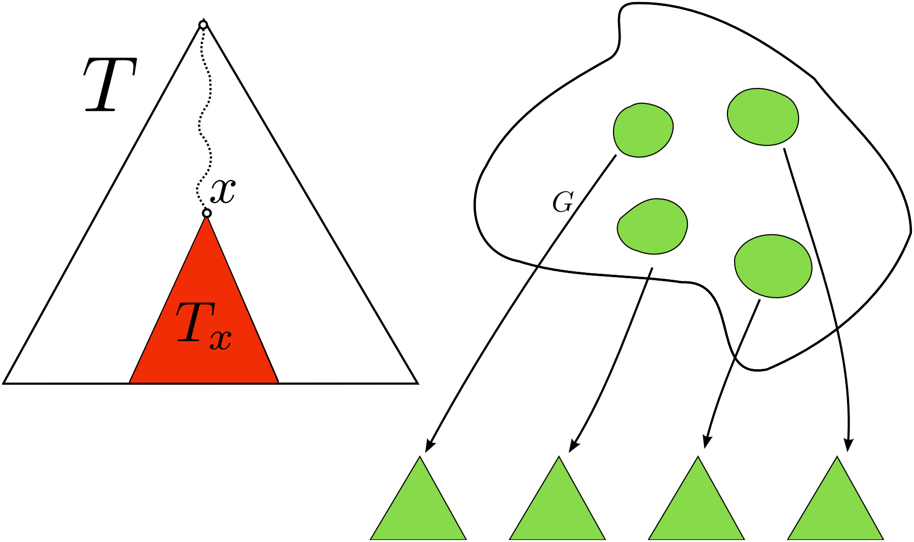
Definition (Nice treedepth decomposition)

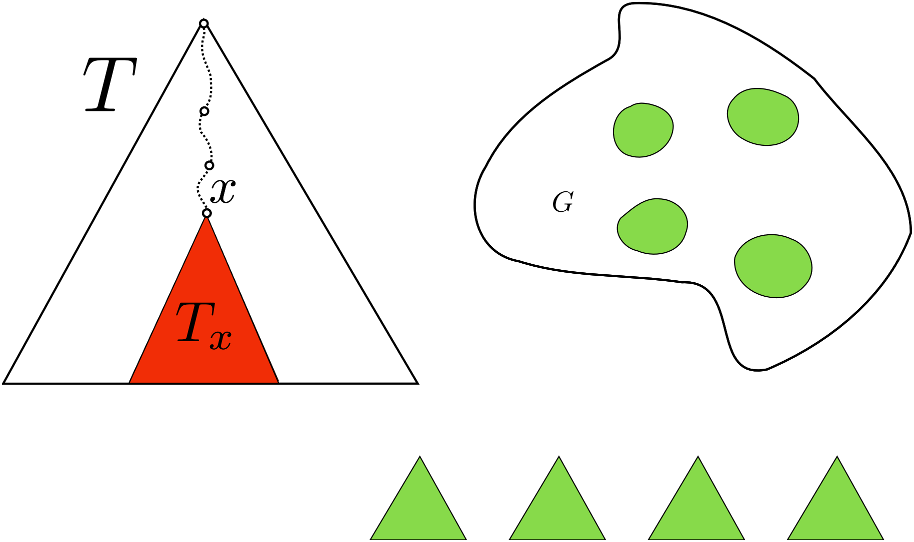
Consider a treedepth decomposition T of G that is not trivially improvable. We say that T is *nice* if for every vertex $x \in V(T)$, the subgraph of G induced by the vertices in T_x is connected.


T
x
T_x
G

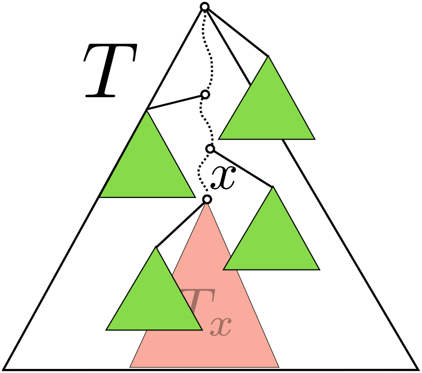


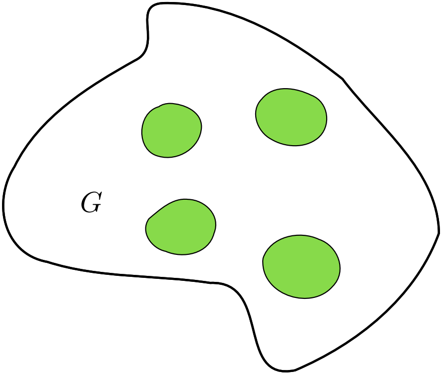
T
x
T_x
G


T
x
T_x
G



T
x
T_x
G

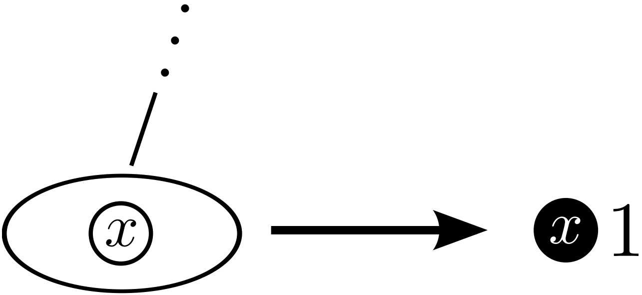

T
x
T_x


G

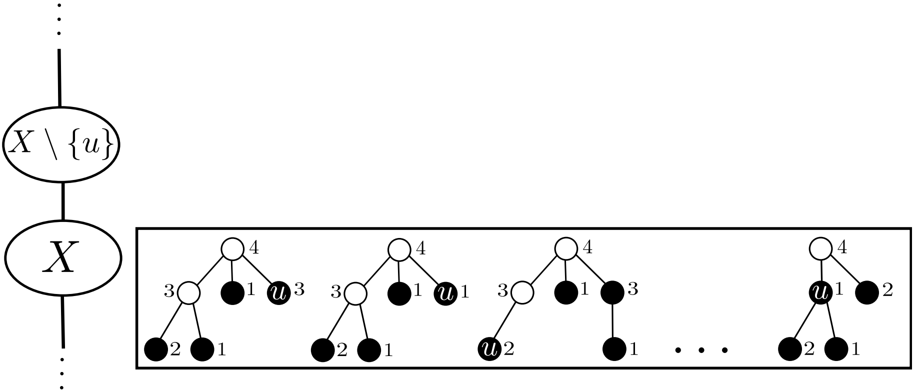
Lemma

For any graph there exists a treedepth decomposition of minimal depth which is nice and non-trivially improvable.

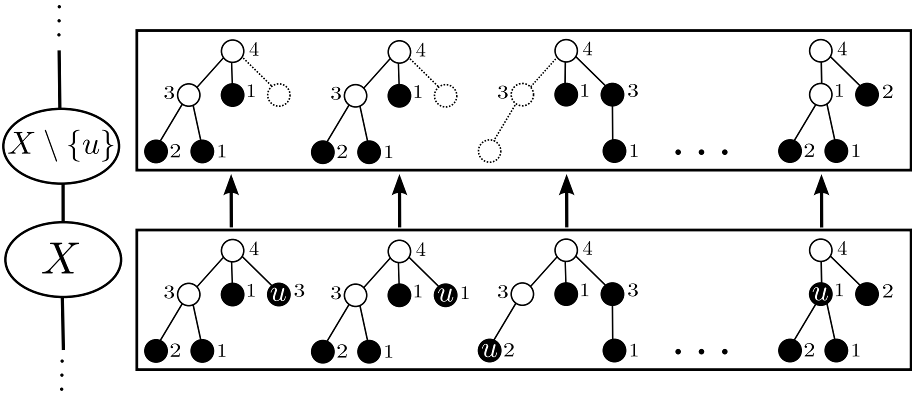
Leaf



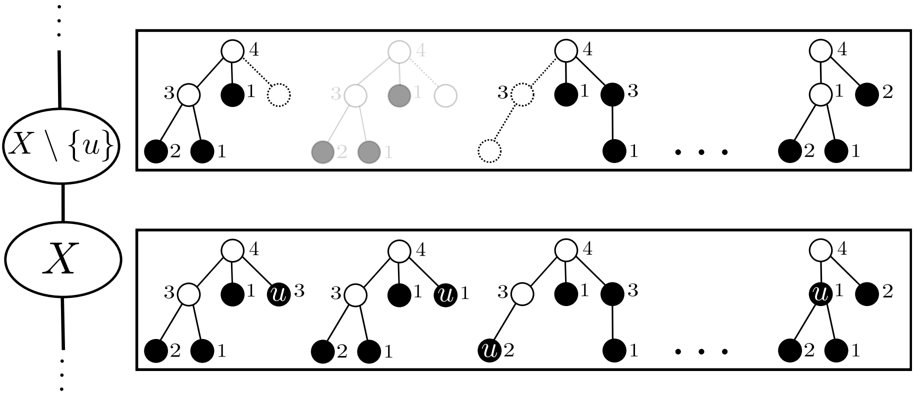
Forget



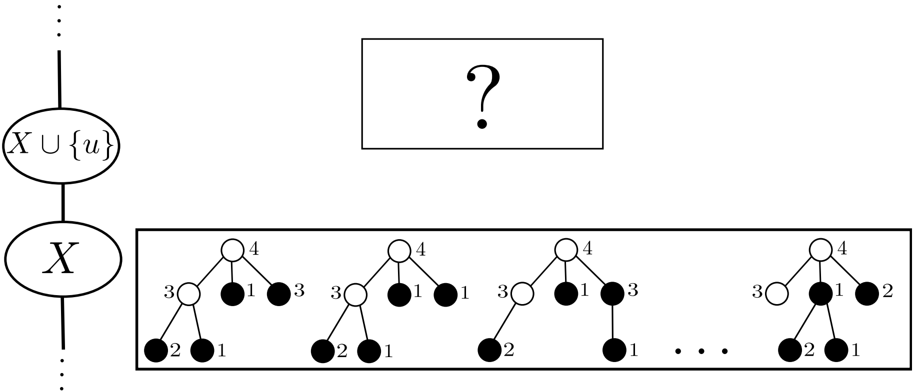
Forget



Forget



Introduce

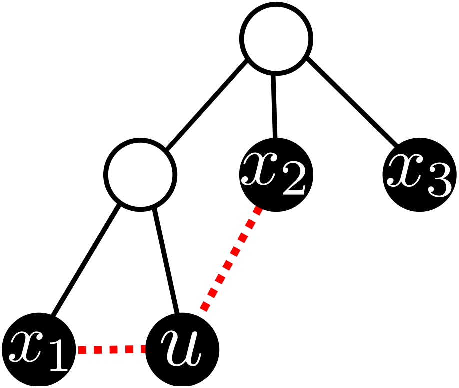


Introduce

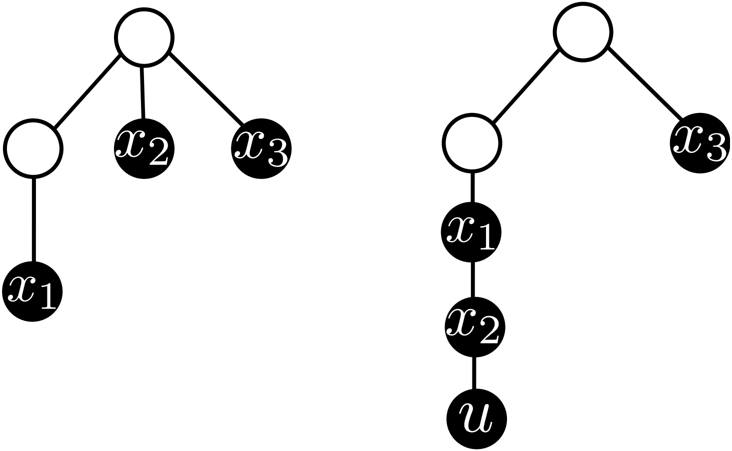
1. We guess every tree T with the following conditions:
   - $X \cup \{u\} \subseteq V(T)$,
   - All leafs are in $X \cup \{u\}$,
   - Height at most t.
2. We keep some, dismiss others.

Introduce

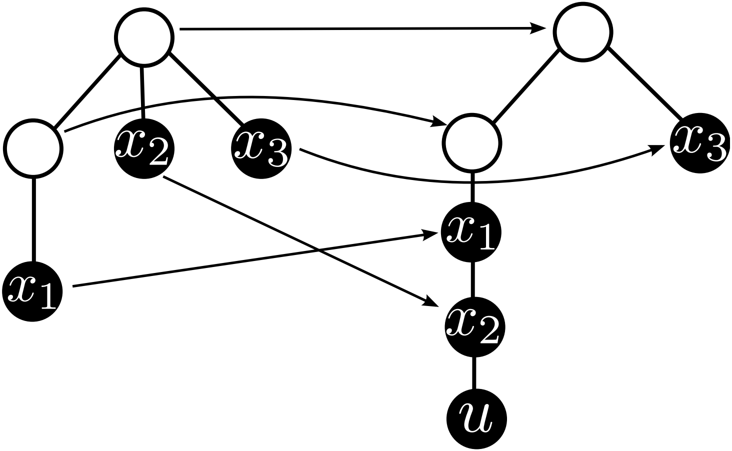
Throw away if edges of X are missing



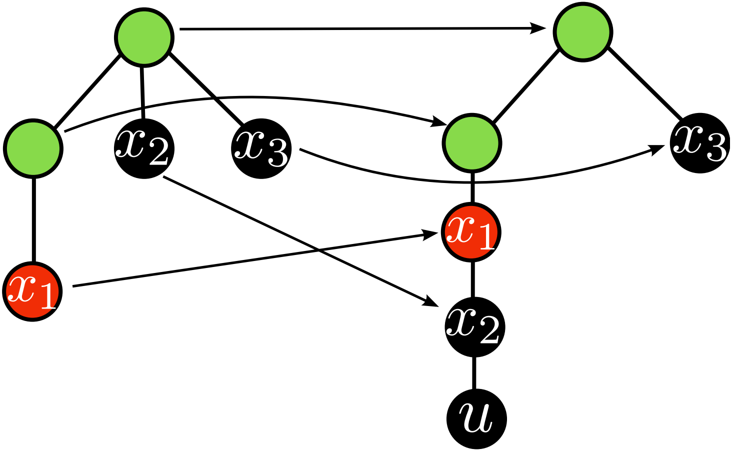
Topological generalization



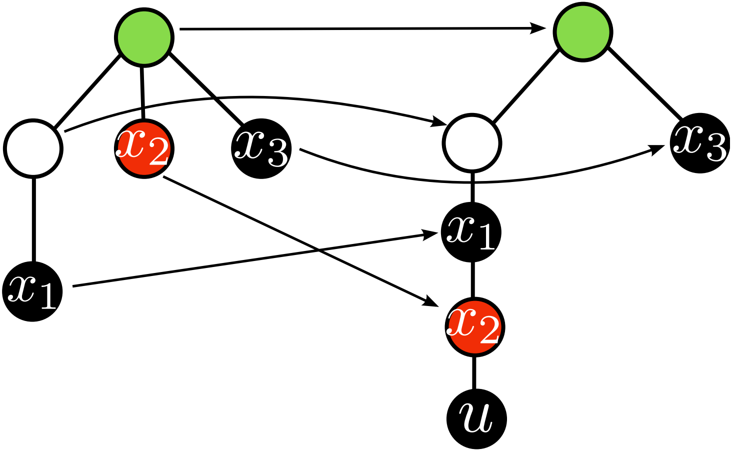
Topological generalization



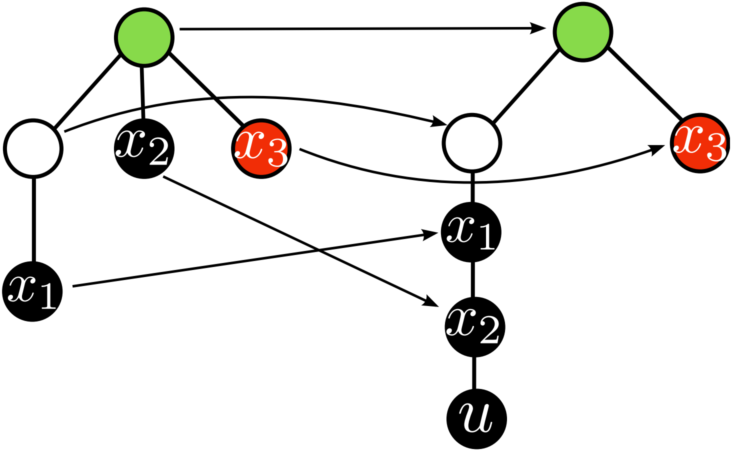
Topological generalization



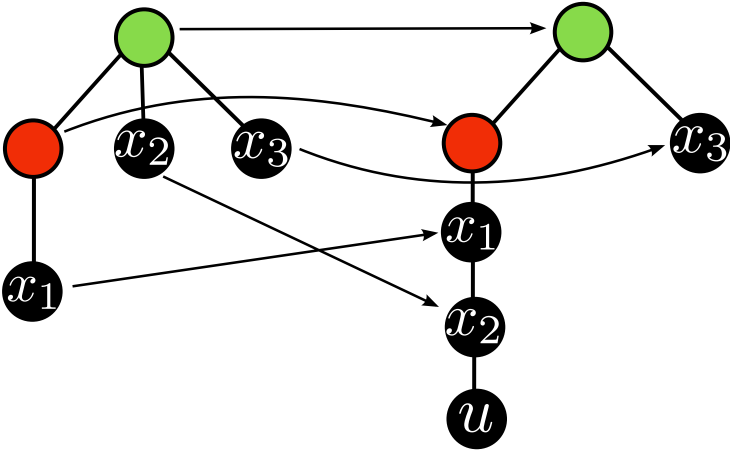
Topological generalization



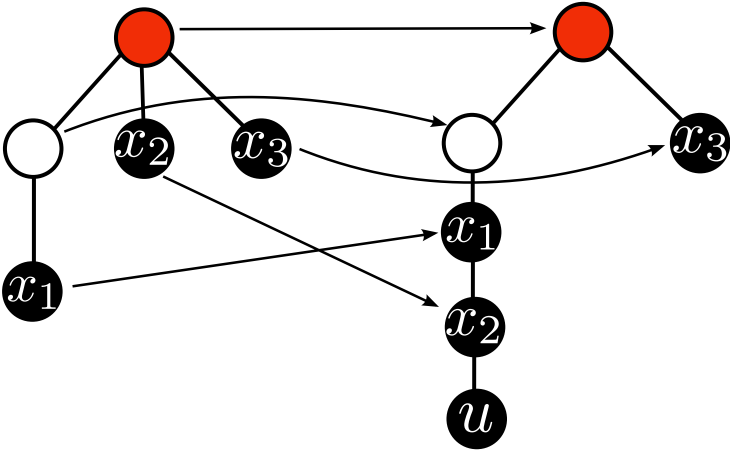
Topological generalization



Topological generalization



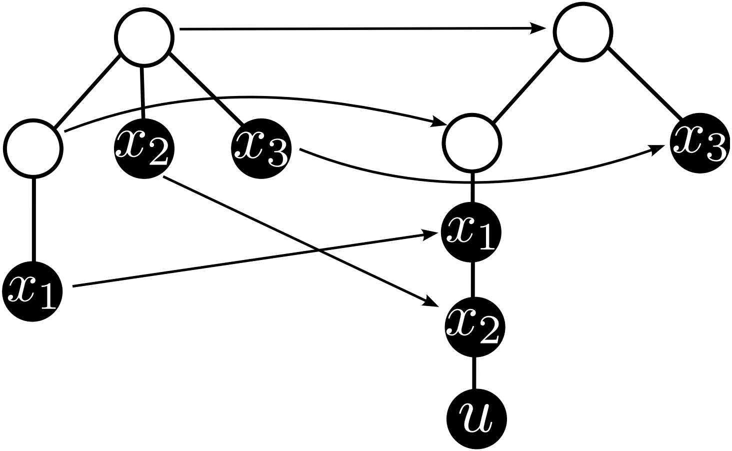
Topological generalization



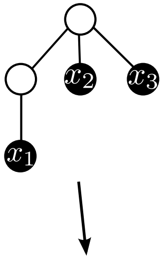
Introduce

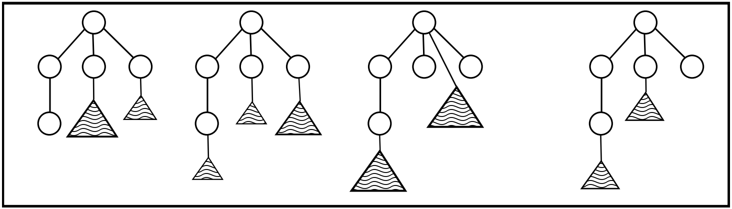
1. Guess tree with certain conditions:
   - $X \cup \{u\} \subseteq V(T)$,
   - All leafs are in $X \cup \{u\}$,
   - Height at most t.
2. If edge missing between nodes of $X \cup \{u\}$, discard.
3. If not a topological generalization of an element in old table, discard.

Why should this work?

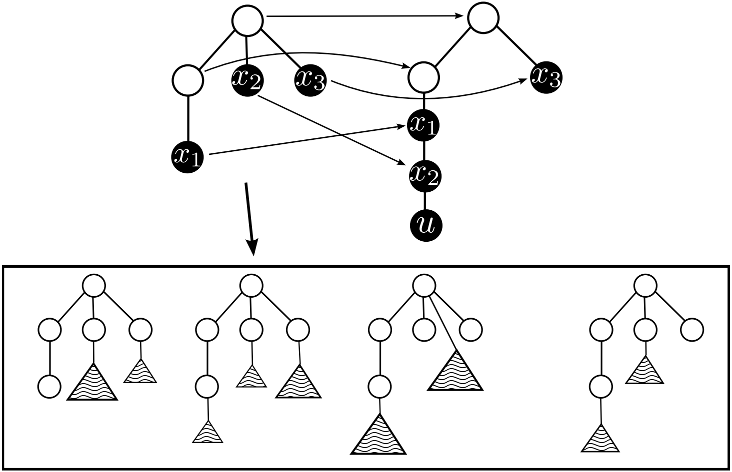


Why should this work?

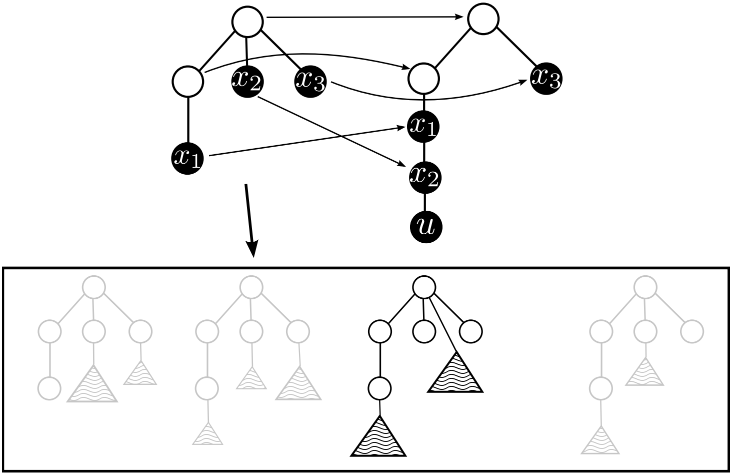




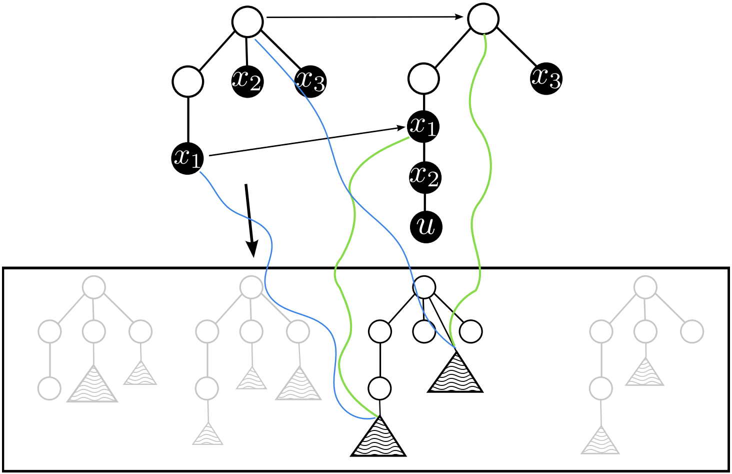
Why should this work?



Why should this work?



Why should this work?



Theorem

Given a graph G with n nodes and a tree decomposition of G of width w the treedepth t of G can be decided in time $2^{O(wt)} \cdot n$.

Simple algorithm

1. Depth-first-search to construct treedepth decomposition T.
2. If depth greater than $2^t - 1$ say NO.
3. Construct path decomposition $\mathcal{P}$ from T of width 2^t.
4. Run algorithm on $\mathcal{P}$.

Theorem

There is a (simple) algorithm to decide if a graph G with n nodes has treedepth t which runs in time $2^{2^{O(t)}} \cdot n$.

Fast algorithm

1. Use single exponential 5-approximation for treewidth[1].
2. Remember $\mathbf{tw}(G) \leq \mathbf{pw}(G) \leq \mathbf{td}(G) - 1$.
3. If width is greater than $5t$ say NO.
4. Else run algorithm on tree decomposition.

Theorem

There is a algorithm to decide if a graph G with n nodes has treedepth t which runs in time $2^{O(t^2)} \cdot n$.

[1] Very useful result by Hans Bodlaender, Pål G. Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov and Michał Pilipczuk

We have seen:

- An algorithm on tree decompositions which runs in time $2^{O(wt)} \cdot n$.
- A (relatively) simple direct algorithm in time $2^{2^{O(t)}} \cdot n$.
- A fast algorithm in time $2^{O(t^2)} \cdot n$.

Related Questions

- Is it sensible to implement?
- Can we approximate treedepth in time $2^{O(t)} \cdot n$.
- Can we find new algorithms on treedepth which are better than working in the path decomposition given by a treedepth decomposition?

Thank you for listening.

Questions?