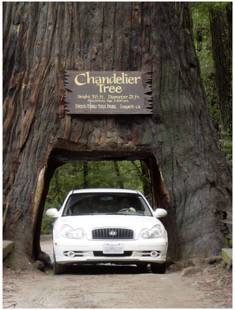
# Computing Treewidth

Tuukka Korhonen

27 February 2025

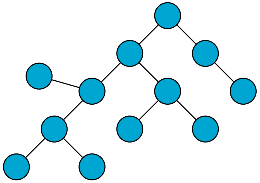
# Plan

1. Introduction to treewidth
2. Background on computing treewidth
3. My work on computing treewidth



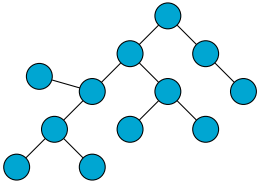
# Treewidth: Introduction

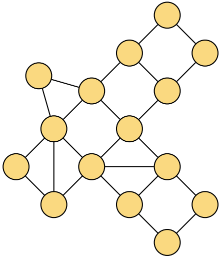
- Many algorithmic problems can be solved more efficiently on trees than on general graphs



# Treewidth: Introduction

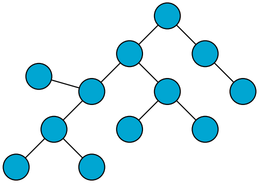
- Many algorithmic problems can be solved more efficiently on trees than on general graphs
- What if a graph is not a tree, but almost?

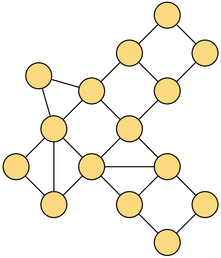




# Treewidth: Introduction

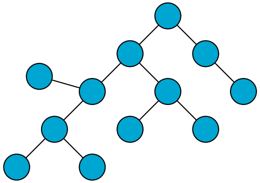
- Many algorithmic problems can be solved more efficiently on trees than on general graphs
- What if a graph is not a tree, but almost?
- The treewidth of a graph

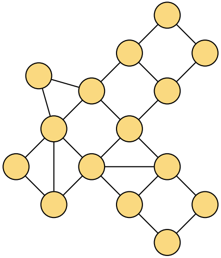




# Treewidth: Introduction

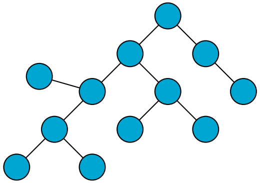
- Many algorithmic problems can be solved more efficiently on trees than on general graphs
- What if a graph is not a tree, but almost?
- The treewidth of a graph
- Trees have treewidth 1

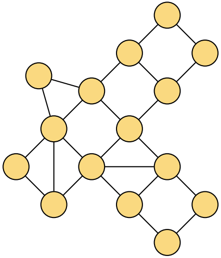




# Treewidth: Introduction

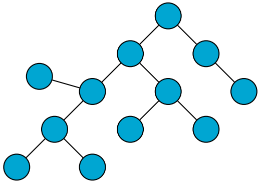
- Many algorithmic problems can be solved more efficiently on trees than on general graphs
- What if a graph is not a tree, but almost?
- The treewidth of a graph
- Trees have treewidth 1
- The example graph has treewidth 2

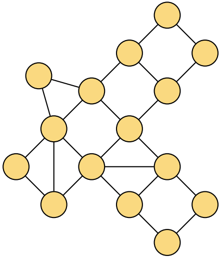




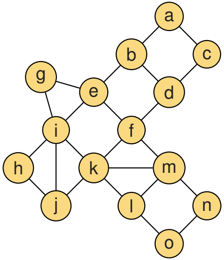
# Treewidth: Introduction

- Many algorithmic problems can be solved more efficiently on trees than on general graphs
- What if a graph is not a tree, but almost?
- The treewidth of a graph
- Trees have treewidth 1
- The example graph has treewidth 2
- Applications in graph algorithms, constraint solving, databases, probabilistic inference, simulating quantum computers...



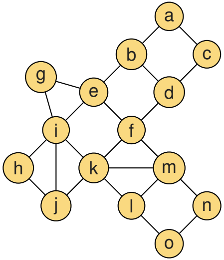


# Treewidth: Definition

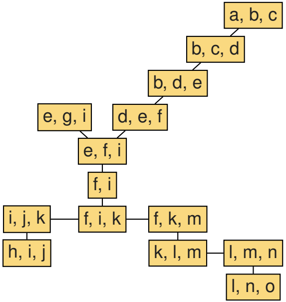



Graph $G$

# Treewidth: Definition

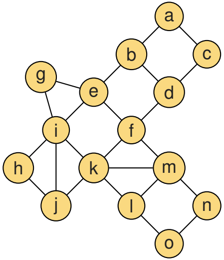



Graph $G$

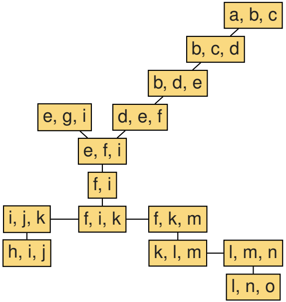



A tree decomposition of $G$

# Treewidth: Definition




Graph $G$

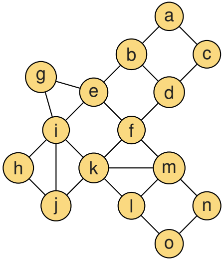



A tree decomposition of $G$

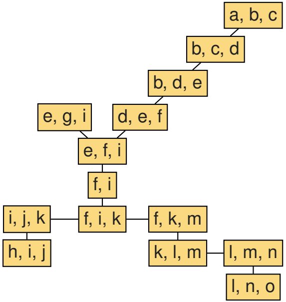
## Tree decomposition:

1. Every vertex should be in a bag
2. Every edge should be in a bag
3. For every vertex $v$, the bags containing $v$ should form a connected subtree

# Treewidth: Definition




Graph $G$



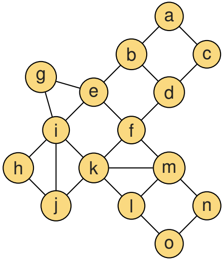

A tree decomposition of $G$
Width = 2

## Tree decomposition:

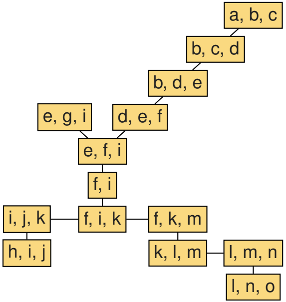
1. Every vertex should be in a bag
2. Every edge should be in a bag
3. For every vertex $v$, the bags containing $v$ should form a connected subtree

- Width = maximum bag size $-1$

# Treewidth: Definition




Graph $G$
Treewidth 2




A tree decomposition of $G$
Width = 2

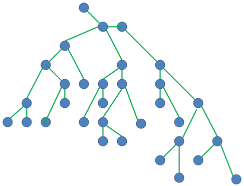
## Tree decomposition:

1. Every vertex should be in a bag
2. Every edge should be in a bag
3. For every vertex $v$, the bags containing $v$ should form a connected subtree

- Width = maximum bag size $-1$
- Treewidth of $G$ = the minimum width of a tree decomposition of $G$

# Treewidth of graphs

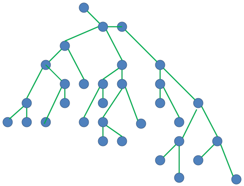
Examples of graphs of small treewidth:



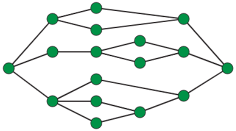
Trees (tw $\leq$ 1)

# Treewidth of graphs

Examples of graphs of small treewidth:



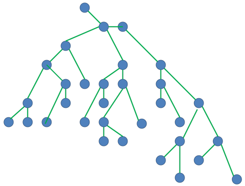
Trees (tw $\leq$ 1)



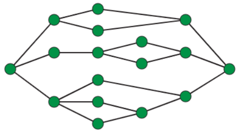
Series-parallel (tw $\leq$ 2)

# Treewidth of graphs

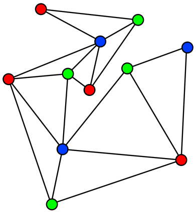
Examples of graphs of small treewidth:



Trees (tw $\leq$ 1)



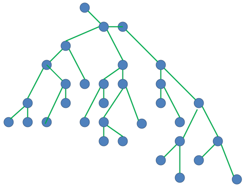
Series-parallel (tw $\leq$ 2)



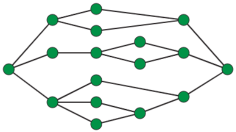
Outerplanar (tw $\leq$ 2)

# Treewidth of graphs

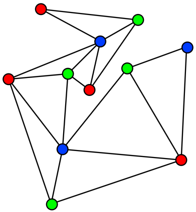
Examples of graphs of small treewidth:



Trees (tw $\leq$ 1)

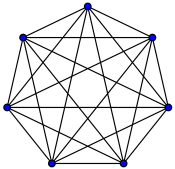


Series-parallel (tw $\leq$ 2)



Outerplanar (tw $\leq$ 2)

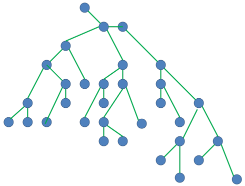
Examples of graphs of large treewidth:



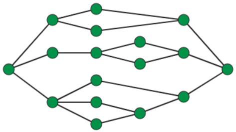
Cliques (tw $= n - 1$)

# Treewidth of graphs

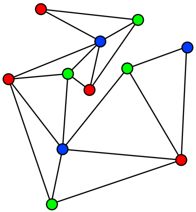
Examples of graphs of small treewidth:



Trees (tw $\leq 1$)

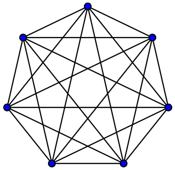


Series-parallel (tw $\leq 2$)

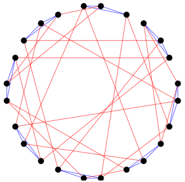


Outerplanar (tw $\leq 2$)

Examples of graphs of large treewidth:



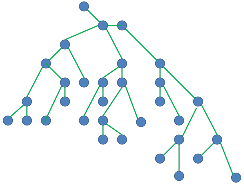
Cliques (tw $= n - 1$)



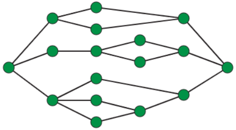
Random graphs (tw $= \Theta(n)$)

# Treewidth of graphs

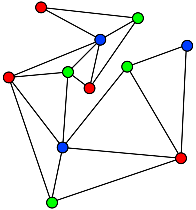
Examples of graphs of small treewidth:



Trees (tw $\leq 1$)

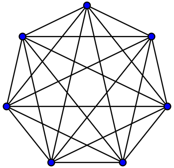


Series-parallel (tw $\leq 2$)

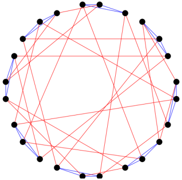


Outerplanar (tw $\leq 2$)

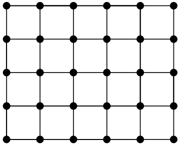
Examples of graphs of large treewidth:



Cliques (tw $= n - 1$)



Random graphs (tw $= \Theta(n)$)

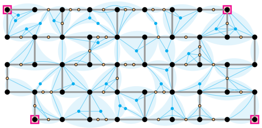


$n \times m$-grids (tw $= \min(n, m)$)

# Treewidth: History and Applications

Treewidth was invented in different formulations by...

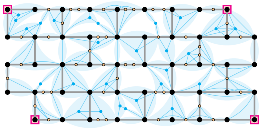
# Treewidth: History and Applications



Treewidth was invented in different formulations by...

- Robertson & Seymour for their Graph Minors series, 1983–2012

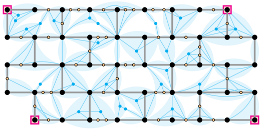
# Treewidth: History and Applications



Treewidth was invented in different formulations by...

- Robertson & Seymour for their Graph Minors series, 1983–2012
- Bertele & Brioschi for solving optimization problems, 1972

# Treewidth: History and Applications



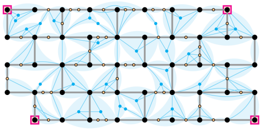
Treewidth was invented in different formulations by...

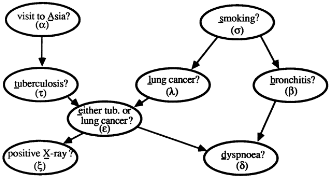
- Robertson & Seymour for their Graph Minors series, 1983–2012
- Bertele & Brioschi for solving optimization problems, 1972
- Arnborg & Proskurowski for solving graph problems, 1989

# Treewidth: History and Applications

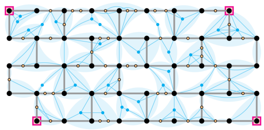
Treewidth was invented in different formulations by...

- Robertson & Seymour for their Graph Minors series, 1983–2012
- Bertele & Brioschi for solving optimization problems, 1972
- Arnborg & Proskurowski for solving graph problems, 1989
- Lauritzen & Spiegelhalter for probabilistic inference, 1988



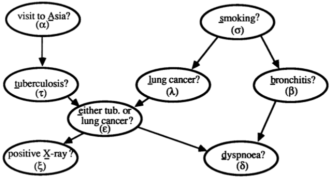


# Treewidth: History and Applications

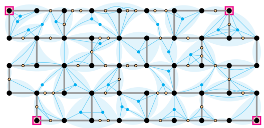


Treewidth was invented in different formulations by...

- Robertson & Seymour for their Graph Minors series, 1983–2012
- Bertele & Brioschi for solving optimization problems, 1972
- Arnborg & Proskurowski for solving graph problems, 1989
- Lauritzen & Spiegelhalter for probabilistic inference, 1988
- Mikkel Thorup for compiler optimization, 1997

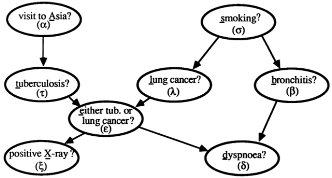


# Treewidth: History and Applications



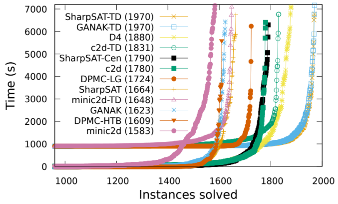
Treewidth was invented in different formulations by...

- Robertson & Seymour for their Graph Minors series, 1983–2012
- Bertele & Brioschi for solving optimization problems, 1972
- Arnborg & Proskurowski for solving graph problems, 1989
- Lauritzen & Spiegelhalter for probabilistic inference, 1988
- Mikkel Thorup for compiler optimization, 1997



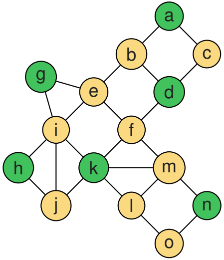

Modern practical applications include at least:

- Probabilistic inference
- Propositional model counting (#SAT)
- Database query evaluation
- Simulating quantum computers



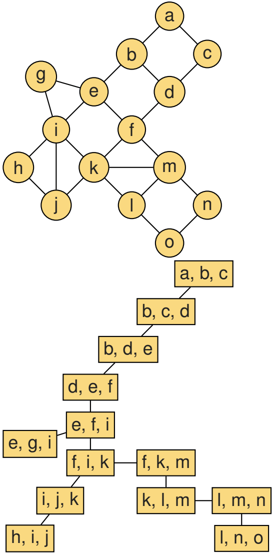
# Treewidth: Example application

- Example: Solving the maximum independent set problem

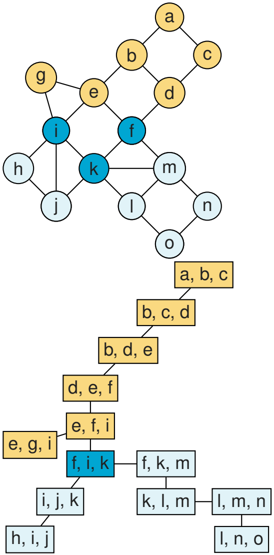


# Treewidth: Example application

- Example: Solving the maximum independent set problem
- $\mathcal{O}(2^k \cdot n)$ time solution, where $k$ width and $n$ the graph size



# Treewidth: Example application




- Example: Solving the maximum independent set problem
- $\mathcal{O}(2^k \cdot n)$ time solution, where $k$ width and $n$ the graph size
- Dynamic programming over states $\text{dp}[t][S]$, where $t$ is a node and $S \subseteq \text{bag}(t)$

# Computing treewidth

Need the tree decomposition!

## Need the tree decomposition!

- Central problem: Compute a tree decomposition of small width if one exists

## Need the tree decomposition!

- Central problem: Compute a tree decomposition of small width if one exists
  - Approximation fine, but slows down the next step

## Need the tree decomposition!

- Central problem: Compute a tree decomposition of small width if one exists
  - Approximation fine, but slows down the next step
  - Running time exponential in $k$ fine

## Need the tree decomposition!

- Central problem: Compute a tree decomposition of small width if one exists
  - Approximation fine, but slows down the next step
  - Running time exponential in $k$ fine
- [Arnborg, Corneil & Proskurowski '87]:
  - Computing treewidth is NP-hard
  - Algorithm with running time $\mathcal{O}(n^{k+2})$

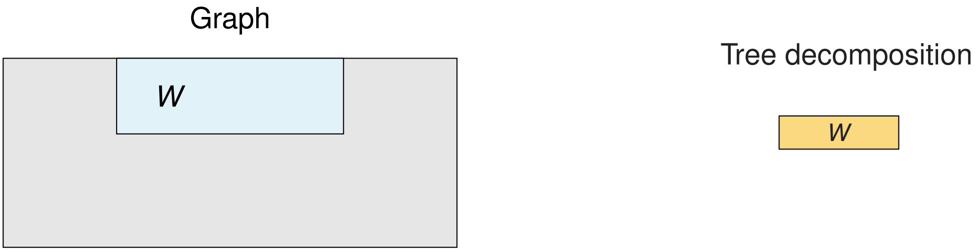
## Need the tree decomposition!

- Central problem: Compute a tree decomposition of small width if one exists
  - Approximation fine, but slows down the next step
  - Running time exponential in $k$ fine
- [Arnborg, Corneil & Proskurowski ’87]:
  - Computing treewidth is NP-hard
  - Algorithm with running time $\mathcal{O}(n^{k+2})$
- [Robertson & Seymour, Graph Minors 13, ’87]:
  - 4-approximation algorithm with running time $\mathcal{O}(3^{3k} \cdot n^2)$
  - Introduced the “top-down” approach for computing tree decompositions

# The Robertson-Seymour top-down approach

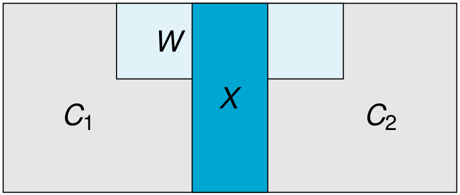
Graph

# The Robertson-Seymour top-down approach

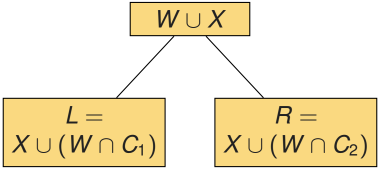


# The Robertson-Seymour top-down approach

Graph

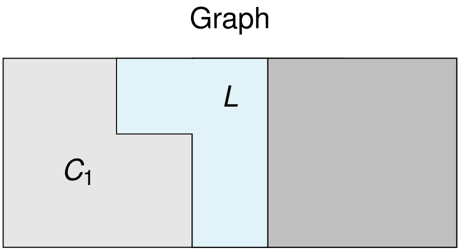


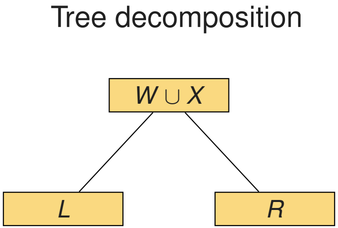

Tree decomposition




Balanced separator $X$ with components $C_1$ and $C_2$

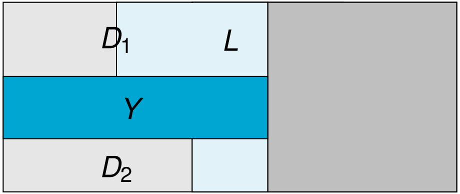
# The Robertson-Seymour top-down approach



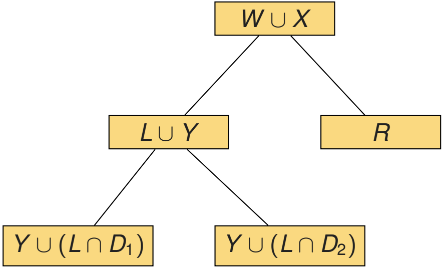
# The Robertson-Seymour top-down approach

Graph



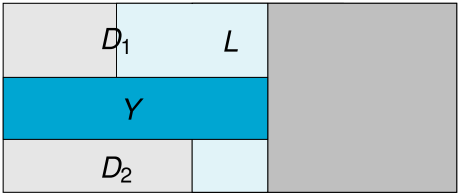

Balanced separator $Y$ with components $D_1$ and $D_2$

Tree decomposition



# The Robertson-Seymour top-down approach

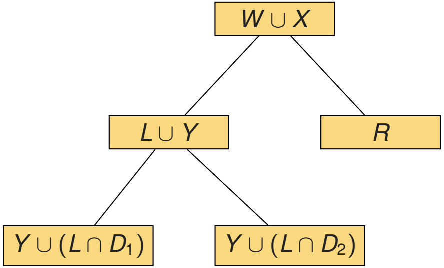
Graph




Balanced separator $Y$ with components $D_1$ and $D_2$

Continue recursively...

Tree decomposition



# Influence of the top-down approach

| Reference | Appx. ratio | Running time |
|---|---|---|
| [Robertson & Seymour '87] | 4 | $\mathcal{O}(3^{3k} \cdot n^2)$ |
| [Matoušek & Thomas '91] | 6 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Lagergren '96] | 8 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Reed '92] | 8 | $k^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '95] | $\mathcal{O}(\log n)$ | $\text{poly}(n)$ |
| [Amir '10] | 4.5 | $\mathcal{O}(2^{3k} \cdot n^2)$ |
| [Amir '10] | $\mathcal{O}(\log k)$ | $\mathcal{O}(k \log k \cdot n^4)$ |
| [Feige, Hajiaghayi & Lee '08] | $\mathcal{O}(\sqrt{\log k})$ | $\text{poly}(n)$ |
| [Bodlaender et al. '16] | 3 | $2^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '16] | 5 | $2^{\mathcal{O}(k)} \cdot n$ |
| [Fomin et al. '18] | $\mathcal{O}(k)$ | $\mathcal{O}(k^7 \cdot n \log n)$ |
| [Belbasi & Fürer '21] | 5 | $\mathcal{O}(2^{7k} \cdot n \log n)$ |

# Influence of the top-down approach

| Reference | Appx. ratio | Running time |
|---|---|---|
| [Robertson & Seymour '87] | 4 | $\mathcal{O}(3^{3k} \cdot n^2)$ |
| [Matoušek & Thomas '91] | 6 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Lagergren '96] | 8 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Reed '92] | 8 | $k^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '95] | $\mathcal{O}(\log n)$ | $\text{poly}(n)$ |
| [Amir '10] | 4.5 | $\mathcal{O}(2^{3k} \cdot n^2)$ |
| [Amir '10] | $\mathcal{O}(\log k)$ | $\mathcal{O}(k \log k \cdot n^4)$ |
| [Feige, Hajiaghayi & Lee '08] | $\mathcal{O}(\sqrt{\log k})$ | $\text{poly}(n)$ |
| [Bodlaender et al. '16] | 3 | $2^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '16] | 5 | $2^{\mathcal{O}(k)} \cdot n$ |
| [Fomin et al. '18] | $\mathcal{O}(k)$ | $\mathcal{O}(k^7 \cdot n \log n)$ |
| [Belbasi & Fürer '21] | 5 | $\mathcal{O}(2^{7k} \cdot n \log n)$ |

- Before 2021, all approximation algorithms for treewidth used this approach

# Influence of the top-down approach

| Reference | Appx. ratio | Running time |
|---|---|---|
| [Robertson & Seymour '87] | 4 | $\mathcal{O}(3^{3k} \cdot n^2)$ |
| [Matoušek & Thomas '91] | 6 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Lagergren '96] | 8 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Reed '92] | 8 | $k^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '95] | $\mathcal{O}(\log n)$ | $\text{poly}(n)$ |
| [Amir '10] | 4.5 | $\mathcal{O}(2^{3k} \cdot n^2)$ |
| [Amir '10] | $\mathcal{O}(\log k)$ | $\mathcal{O}(k \log k \cdot n^4)$ |
| [Feige, Hajiaghayi & Lee '08] | $\mathcal{O}(\sqrt{\log k})$ | $\text{poly}(n)$ |
| [Bodlaender et al. '16] | 3 | $2^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '16] | 5 | $2^{\mathcal{O}(k)} \cdot n$ |
| [Fomin et al. '18] | $\mathcal{O}(k)$ | $\mathcal{O}(k^7 \cdot n \log n)$ |
| [Belbasi & Fürer '21] | 5 | $\mathcal{O}(2^{7k} \cdot n \log n)$ |

- Before 2021, all approximation algorithms for treewidth used this approach
- Barrier at approximation ratio 3

# Influence of the top-down approach

| Reference | Appx. ratio | Running time |
|---|---|---|
| [Robertson & Seymour '87] | 4 | $\mathcal{O}(3^{3k} \cdot n^2)$ |
| [Matoušek & Thomas '91] | 6 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Lagergren '96] | 8 | $k^{\mathcal{O}(k)} \cdot n \log^2 n$ |
| [Reed '92] | 8 | $k^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '95] | $\mathcal{O}(\log n)$ | $\text{poly}(n)$ |
| [Amir '10] | 4.5 | $\mathcal{O}(2^{3k} \cdot n^2)$ |
| [Amir '10] | $\mathcal{O}(\log k)$ | $\mathcal{O}(k \log k \cdot n^4)$ |
| [Feige, Hajiaghayi & Lee '08] | $\mathcal{O}(\sqrt{\log k})$ | $\text{poly}(n)$ |
| [Bodlaender et al. '16] | 3 | $2^{\mathcal{O}(k)} \cdot n \log n$ |
| [Bodlaender et al. '16] | 5 | $2^{\mathcal{O}(k)} \cdot n$ |
| [Fomin et al. '18] | $\mathcal{O}(k)$ | $\mathcal{O}(k^7 \cdot n \log n)$ |
| [Belbasi & Fürer '21] | 5 | $\mathcal{O}(2^{7k} \cdot n \log n)$ |

- Before 2021, all approximation algorithms for treewidth used this approach
- Barrier at approximation ratio 3
- Hard to implement in linear time

# My contribution

- State-of-the-art before 2021:

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]
  - 3-approximation in $2^{\mathcal{O}(k)} \cdot n \log n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]
  - 3-approximation in $2^{\mathcal{O}(k)} \cdot n \log n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - 5-approximation in $2^{\mathcal{O}(k)} \cdot n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]
  - 3-approximation in $2^{\mathcal{O}(k)} \cdot n \log n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - 5-approximation in $2^{\mathcal{O}(k)} \cdot n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $\mathcal{O}(\sqrt{\log k})$-approximation in $\text{poly}(n)$ time [Feige, Hajiaghayi & Lee '08]

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]
  - 3-approximation in $2^{\mathcal{O}(k)} \cdot n \log n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - 5-approximation in $2^{\mathcal{O}(k)} \cdot n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $\mathcal{O}(\sqrt{\log k})$-approximation in $\text{poly}(n)$ time [Feige, Hajiaghayi & Lee '08]

**Theorem (Korhonen '21)**

There is a 2-approximation algorithm for treewidth with running time $2^{\mathcal{O}(k)} \cdot n$

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]
  - 3-approximation in $2^{\mathcal{O}(k)} \cdot n \log n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - 5-approximation in $2^{\mathcal{O}(k)} \cdot n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $\mathcal{O}(\sqrt{\log k})$-approximation in $\text{poly}(n)$ time [Feige, Hajiaghayi & Lee '08]

**Theorem (Korhonen '21)**

There is a 2-approximation algorithm for treewidth with running time $2^{\mathcal{O}(k)} \cdot n$

- A completely new approach

# My contribution

- State-of-the-art before 2021:
  - exact in $2^{\mathcal{O}(k^3)} \cdot n$ time [Bodlaender '96]
  - 3-approximation in $2^{\mathcal{O}(k)} \cdot n \log n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - 5-approximation in $2^{\mathcal{O}(k)} \cdot n$ time [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $\mathcal{O}(\sqrt{\log k})$-approximation in poly($n$) time [Feige, Hajiaghayi & Lee '08]

**Theorem (Korhonen '21)**

There is a 2-approximation algorithm for treewidth with running time $2^{\mathcal{O}(k)} \cdot n$

- A completely new approach
  - Inspired by the proofs of [Thomas '90] and [Bellenbaum & Diestel '02] on "lean tree decompositions"

# The 2-approximation algorithm

# Outline

# Outline

By a self-reduction technique of [Bodlaender '96] we can focus on giving an improver algorithm:

**Input:** An graph $G$ and a tree decomposition $T$ of $G$ of width $w$

**Output:** A tree decomposition of $G$ of width $\leq w - 1$ or the conclusion that $w \leq 2 \cdot \text{tw}(G) + 1$

**Running time:** $2^{\mathcal{O}(w)} n$

# Outline

By a self-reduction technique of [Bodlaender '96] we can focus on giving an improver algorithm:

**Input:** An graph $G$ and a tree decomposition $T$ of $G$ of width $w$

**Output:** A tree decomposition of $G$ of width $\leq w - 1$ or the conclusion that $w \leq 2 \cdot \text{tw}(G) + 1$

**Running time:** $2^{\mathcal{O}(w)} n$

1. If $w > 2 \cdot \text{tw}(G) + 1$ then $T$ can be improved by a certain improvement operation
   - Decreases the number of largest bags and does not increase the width

# Outline

By a self-reduction technique of [Bodlaender '96] we can focus on giving an improver algorithm:

**Input:** An graph $G$ and a tree decomposition $T$ of $G$ of width $w$

**Output:** A tree decomposition of $G$ of width $\leq w - 1$ or the conclusion that $w \leq 2 \cdot \text{tw}(G) + 1$

**Running time:** $2^{\mathcal{O}(w)} n$

1. If $w > 2 \cdot \text{tw}(G) + 1$ then $T$ can be improved by a certain improvement operation
   - Decreases the number of largest bags and does not increase the width
2. To improve the width by one, $\Omega(n)$ improvement operations may be needed

# Outline

By a self-reduction technique of [Bodlaender '96] we can focus on giving an improver algorithm:

**Input:** An graph $G$ and a tree decomposition $T$ of $G$ of width $w$

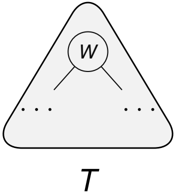
**Output:** A tree decomposition of $G$ of width $\leq w - 1$ or the conclusion that $w \leq 2 \cdot \text{tw}(G) + 1$

**Running time:** $2^{\mathcal{O}(w)} n$

1. If $w > 2 \cdot \text{tw}(G) + 1$ then $T$ can be improved by a certain improvement operation
   - Decreases the number of largest bags and does not increase the width
2. To improve the width by one, $\Omega(n)$ improvement operations may be needed
   - Efficient implementation by amortized analysis of the improvements and dynamic programming over the tree decomposition

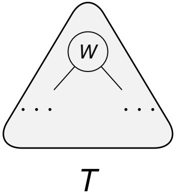
# The improvement operation

- Let $W$ be the largest bag



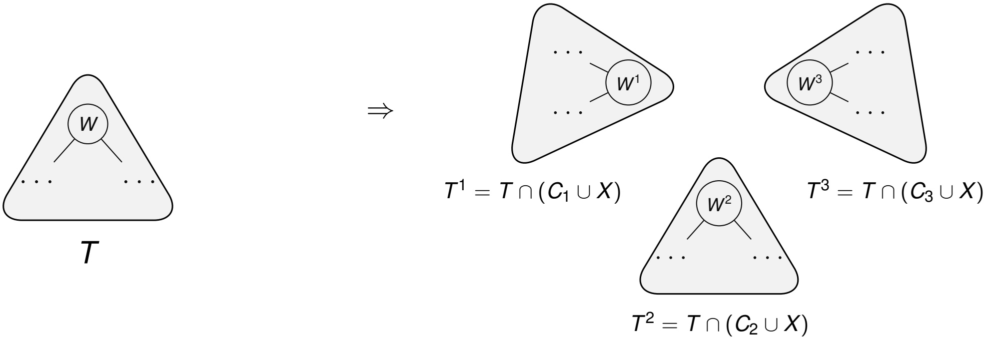
# The improvement operation

- Let $W$ be the largest bag
- Take a separator $X$ of $G$ with a partition $(X, C_1, C_2, C_3)$ of $V(G)$, s.t. $|X \cup (W \cap C_i)| < |W|$ for all $i$



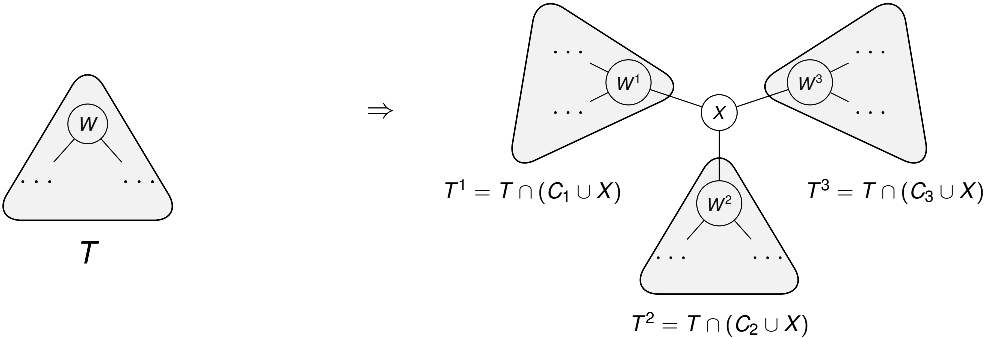
# The improvement operation

- Let $W$ be the largest bag
- Take a separator $X$ of $G$ with a partition $(X, C_1, C_2, C_3)$ of $V(G)$, s.t. $|X \cup (W \cap C_i)| < |W|$ for all $i$
- For each $i$, obtain a tree decomposition $T^i = T \cap (C_i \cup X)$ by setting $B^i = B \cap (C_i \cup X)$ for each bag $B$



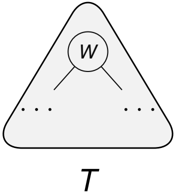
# The improvement operation

- Let $W$ be the largest bag
- Take a separator $X$ of $G$ with a partition $(X, C_1, C_2, C_3)$ of $V(G)$, s.t. $|X \cup (W \cap C_i)| < |W|$ for all $i$
- For each $i$, obtain a tree decomposition $T^i = T \cap (C_i \cup X)$ by setting $B^i = B \cap (C_i \cup X)$ for each bag $B$
- The following is almost a tree decomposition of $G$:

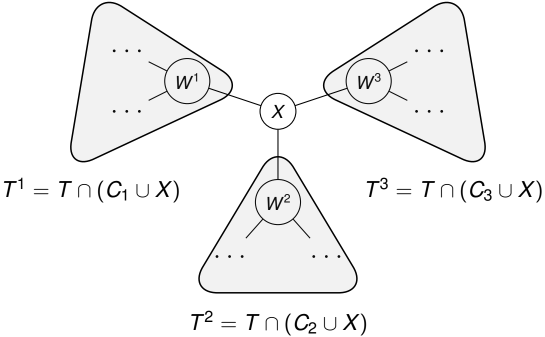


## The improvement operation

- Let $W$ be the largest bag
- Take a separator $X$ of $G$ with a partition $(X, C_1, C_2, C_3)$ of $V(G)$, s.t. $|X \cup (W \cap C_i)| < |W|$ for all $i$
- For each $i$, obtain a tree decomposition $T^i = T \cap (C_i \cup X)$ by setting $B^i = B \cap (C_i \cup X)$ for each bag $B$
- The following is almost a tree decomposition of $G$:




$\Rightarrow$




Except that vertices in $X$ may violate the connectedness condition

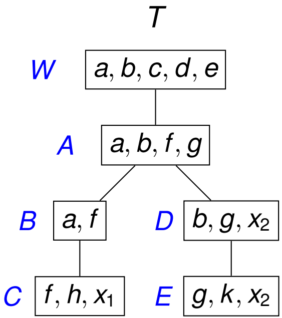
# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

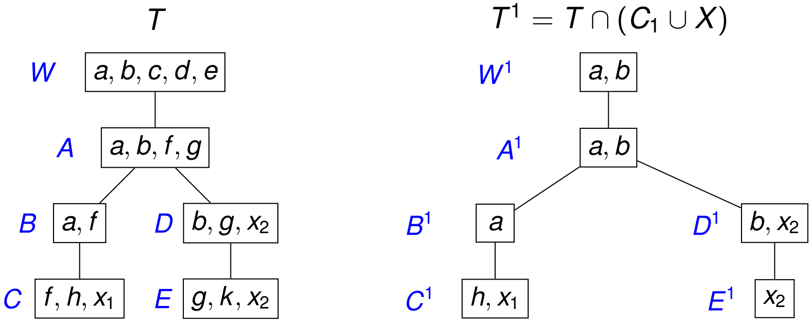
Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:



# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

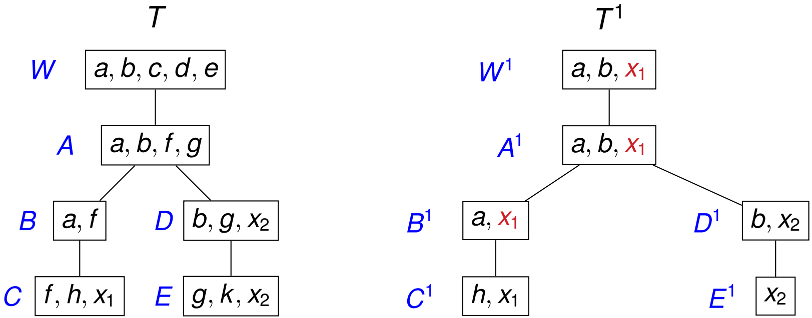
Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:



# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:

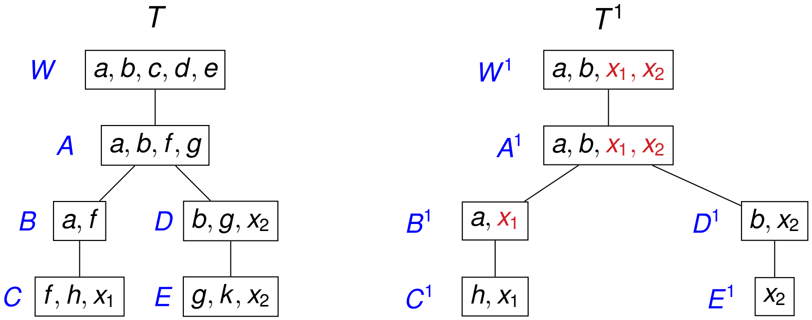



- Insert $x_1$ to $B^1$, $A^1$, and $W^1$

# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:

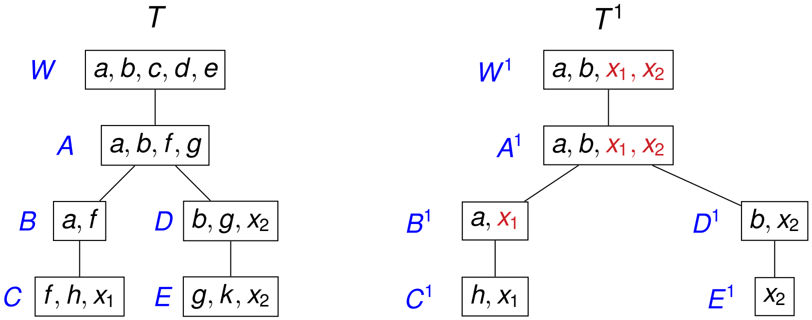



- Insert $x_1$ to $B^1$, $A^1$, and $W^1$
- Insert $x_2$ to $A^1$ and $W^1$

# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:

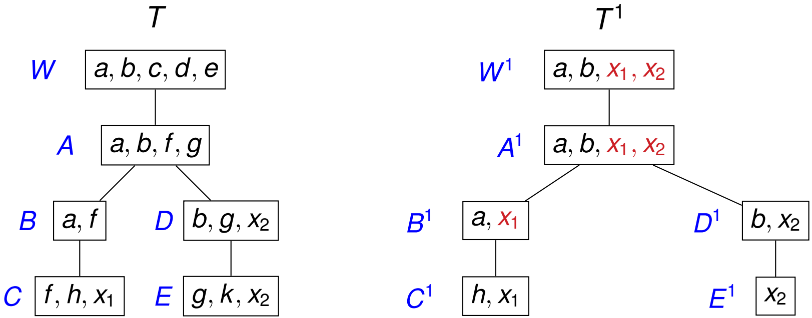



- Insert $x_1$ to $B^1$, $A^1$, and $W^1$
- Insert $x_2$ to $A^1$ and $W^1$
- Now $X \subseteq W^1$ and $T^1$ satisfies the connectedness condition

# Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of $X$ to bags

Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:




- Insert $x_1$ to $B^1$, $A^1$, and $W^1$
- Insert $x_2$ to $A^1$ and $W^1$
- Now $X \subseteq W^1$ and $T^1$ satisfies the connectedness condition

$\Rightarrow$ The whole construction satisfies the connectedness condition

# The main insight

**Definition (Good separation)**

A separation $(X, C_1, C_2, C_3)$ is a *good separation* if (1) $|X \cup (W \cap C_i)| < |W|$ for all $i$, and (2) among those, we minimize $|X|$.

# The main insight

**Definition (Good separation)**

A separation $(X, C_1, C_2, C_3)$ is a *good separation* if (1) $|X \cup (W \cap C_i)| < |W|$ for all $i$, and (2) among those, we minimize $|X|$.

**Lemma**

If $(X, C_1, C_2, C_3)$ is a good separation, then $|B^i| \leq |B|$ for all bags $B$ and all $i$

# The main insight

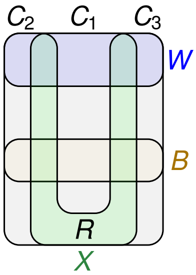
## Definition (Good separation)

A separation $(X, C_1, C_2, C_3)$ is a *good separation* if (1) $|X \cup (W \cap C_i)| < |W|$ for all $i$, and (2) among those, we minimize $|X|$.

## Lemma

If $(X, C_1, C_2, C_3)$ is a good separation, then $|B^i| \leq |B|$ for all bags $B$ and all $i$

- Suppose that $i = 1$ and $|B^1| > |B|$



## The main insight

### Definition (Good separation)

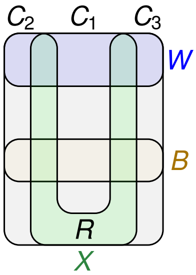
A separation $(X, C_1, C_2, C_3)$ is a *good separation* if (1) $|X \cup (W \cap C_i)| < |W|$ for all $i$, and (2) among those, we minimize $|X|$.

### Lemma

If $(X, C_1, C_2, C_3)$ is a good separation, then $|B^i| \leq |B|$ for all bags $B$ and all $i$

- Suppose that $i = 1$ and $|B^1| > |B|$

$\Rightarrow$ $|R| > |B \cap (C_2 \cup C_3)|$



# The main insight

**Definition (Good separation)**

A separation $(X, C_1, C_2, C_3)$ is a *good separation* if (1) $|X \cup (W \cap C_i)| < |W|$ for all $i$, and (2) among those, we minimize $|X|$.

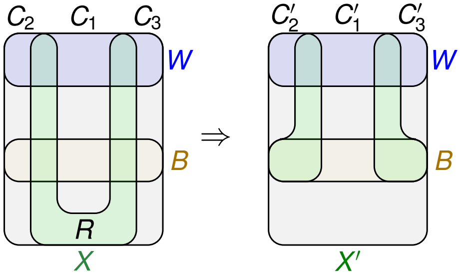
**Lemma**

If $(X, C_1, C_2, C_3)$ is a good separation, then $|B^i| \leq |B|$ for all bags $B$ and all $i$

- Suppose that $i = 1$ and $|B^1| > |B|$

$\Rightarrow$ $|R| > |B \cap (C_2 \cup C_3)|$

- Take a separation with $X' = (X \setminus R) \cup (B \cap (C_2 \cup C_3))$



# The main insight

## Definition (Good separation)

A separation $(X, C_1, C_2, C_3)$ is a *good separation* if (1) $|X \cup (W \cap C_i)| < |W|$ for all $i$, and (2) among those, we minimize $|X|$.

## Lemma

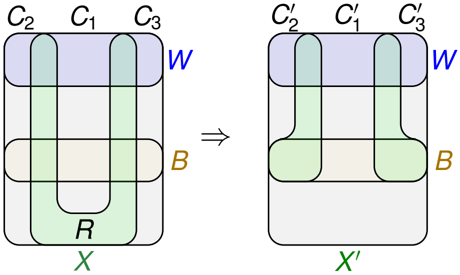
If $(X, C_1, C_2, C_3)$ is a good separation, then $|B^i| \leq |B|$ for all bags $B$ and all $i$

- Suppose that $i = 1$ and $|B^1| > |B|$

$\Rightarrow$ $|R| > |B \cap (C_2 \cup C_3)|$

- Take a separation with $X' = (X \setminus R) \cup (B \cap (C_2 \cup C_3))$

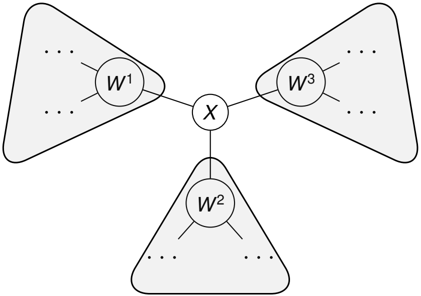
$\Rightarrow$ $|X'| < |X|$ so this contradicts the minimality



# Outlook

We have shown:

- Root bag $W$ replaced by four smaller bags, $W^1$, $W^2$, $W^3$, and $X$
- Width did not increase



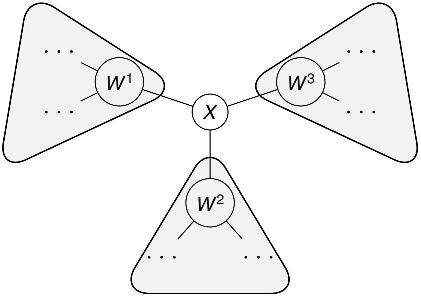
# Outlook

We have shown:

- Root bag $W$ replaced by four smaller bags, $W^1$, $W^2$, $W^3$, and $X$
- Width did not increase

Should also show:

- Number of bags of size $|W|$ decreases



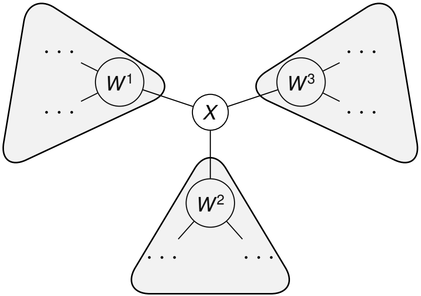
# Outlook

We have shown:

- Root bag $W$ replaced by four smaller bags, $W^1$, $W^2$, $W^3$, and $X$
- Width did not increase

Should also show:

- Number of bags of size $|W|$ decreases
- Good separations can be found efficiently



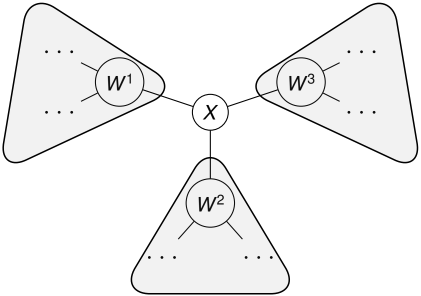
# Outlook

We have shown:

- Root bag $W$ replaced by four smaller bags, $W^1$, $W^2$, $W^3$, and $X$
- Width did not increase

Should also show:

- Number of bags of size $|W|$ decreases
- Good separations can be found efficiently
- $n$ iterations of improvements can be implemented in total $2^{\mathcal{O}(w)} n$ time



# Final remarks

# Follow-up work

Treewidth 2-approximation
[Korhonen, FOCS'21]

# Follow-up work

Treewidth 2-approximation
[Korhonen, FOCS'21]

Improved exact treewidth
[Korhonen & Lokshtanov, STOC'23]

# Follow-up work

Treewidth 2-approximation
[Korhonen, FOCS'21]

↓

Improved exact treewidth
[Korhonen & Lokshtanov, STOC'23]

↓

Dynamic treewidth
[Korhonen, Majewski, Nadara, Pilipczuk & Sokołowski, FOCS'23]

# Follow-up work

Treewidth 2-approximation
[Korhonen, FOCS'21]

↓

Improved exact treewidth
[Korhonen & Lokshtanov, STOC'23]

↓

Dynamic treewidth
[Korhonen, Majewski, Nadara, Pilipczuk & Sokołowski, FOCS'23]

↓

Graph minors in $f(k) \cdot m^{1+\varepsilon}$ time
[Korhonen, Pilipczuk & Stamoulis, FOCS'24]

# Follow-up work

Treewidth 2-approximation
[Korhonen, FOCS'21]

Improved exact treewidth
[Korhonen & Lokshtanov, STOC'23]

Rankwidth in $f(k) \cdot n^2$ time
[Fomin & Korhonen, STOC'22]

Dynamic treewidth
[Korhonen, Majewski, Nadara, Pilipczuk & Sokołowski, FOCS'23]

Graph minors in $f(k) \cdot m^{1+\varepsilon}$ time
[Korhonen, Pilipczuk & Stamoulis, FOCS'24]

# Follow-up work

Treewidth 2-approximation
[Korhonen, FOCS'21]

Improved exact treewidth
[Korhonen & Lokshtanov, STOC'23]

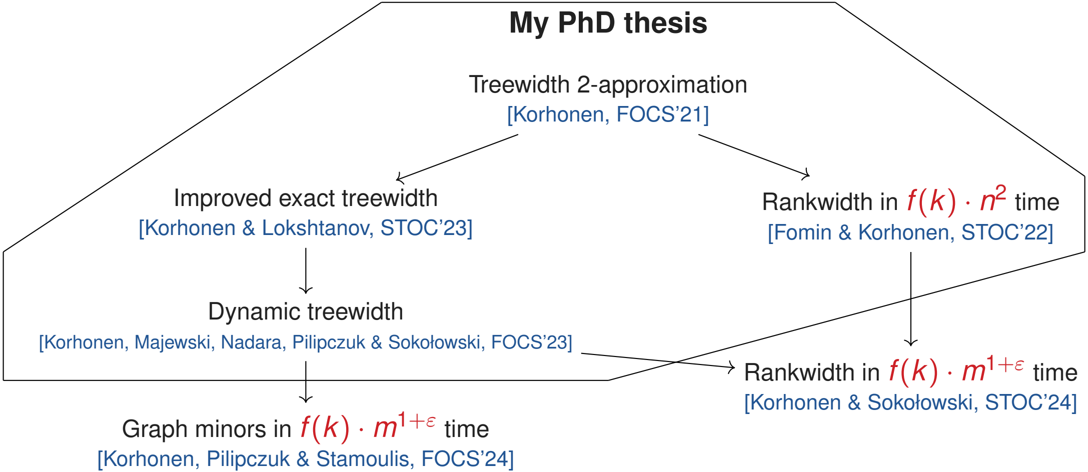
Rankwidth in $f(k) \cdot n^2$ time
[Fomin & Korhonen, STOC'22]

Dynamic treewidth
[Korhonen, Majewski, Nadara, Pilipczuk & Sokołowski, FOCS'23]

Rankwidth in $f(k) \cdot m^{1+\varepsilon}$ time
[Korhonen & Sokołowski, STOC'24]

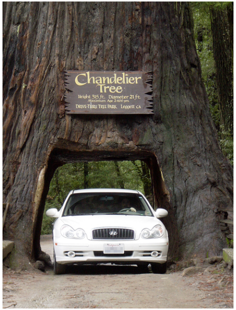
Graph minors in $f(k) \cdot m^{1+\varepsilon}$ time
[Korhonen, Pilipczuk & Stamoulis, FOCS'24]

# Follow-up work



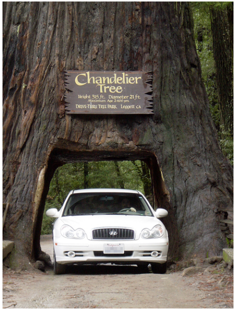
# Conclusion

- Treewidth has applications in many areas of computer science



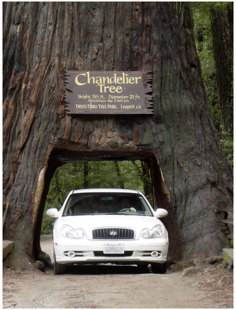
# Conclusion

- Treewidth has applications in many areas of computer science
- My PhD thesis: New paradigm in treewidth computing



# Conclusion

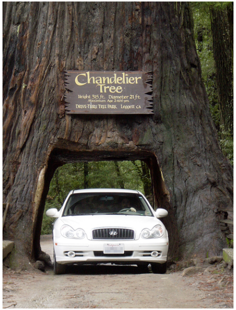
- Treewidth has applications in many areas of computer science
- My PhD thesis: New paradigm in treewidth computing
  - Approximate, exact, and dynamic computing of treewidth, also rankwidth and graph minors



# Conclusion

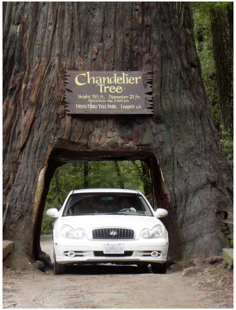
- Treewidth has applications in many areas of computer science
- My PhD thesis: New paradigm in treewidth computing
  - Approximate, exact, and dynamic computing of treewidth, also rankwidth and graph minors

Future:



# Conclusion

- Treewidth has applications in many areas of computer science
- My PhD thesis: New paradigm in treewidth computing
  - Approximate, exact, and dynamic computing of treewidth, also rankwidth and graph minors

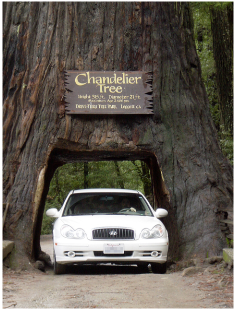



Future:

- End goal: $2^{\mathcal{O}(k)} n$ time exact algorithm for treewidth?

# Conclusion

- Treewidth has applications in many areas of computer science
- My PhD thesis: New paradigm in treewidth computing
  - Approximate, exact, and dynamic computing of treewidth, also rankwidth and graph minors

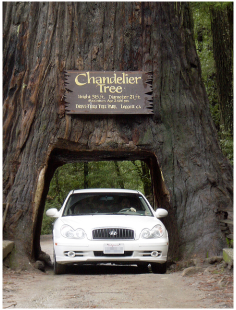



Future:

- End goal: $2^{\mathcal{O}(k)} n$ time exact algorithm for treewidth?
- Currently working on:
  - Improved dynamic treewidth
  - Preprocessing for treewidth

# Conclusion

- Treewidth has applications in many areas of computer science
- My PhD thesis: New paradigm in treewidth computing
  - Approximate, exact, and dynamic computing of treewidth, also rankwidth and graph minors




Future:

- End goal: $2^{\mathcal{O}(k)} n$ time exact algorithm for treewidth?
- Currently working on:
  - Improved dynamic treewidth
  - Preprocessing for treewidth

Thank you!