Computing Treewidth

Tuukka Korhonen

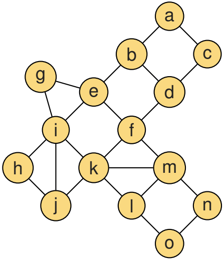
UNIVERSITY OF BERGEN

FPT Fest 2023 in the honor of Mike Fellows

15 June 2023

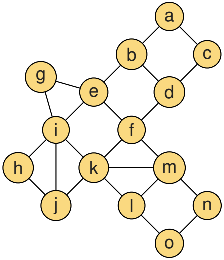
Treewidth

- Measures how close a graph is to a tree



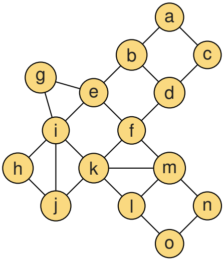
Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1



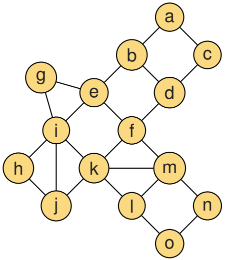
Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2



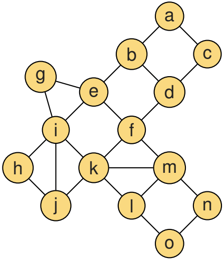
Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n



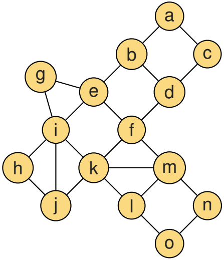
Treewidth

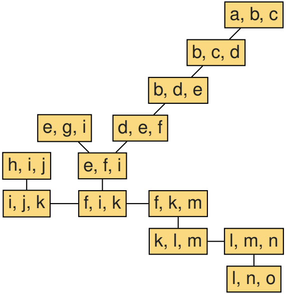
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$



Treewidth

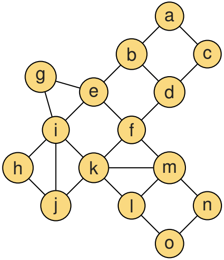
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition

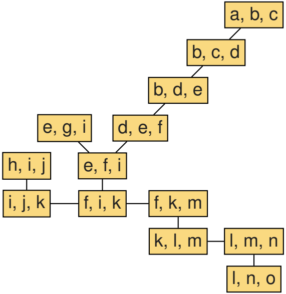




Treewidth

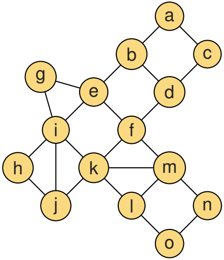
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:

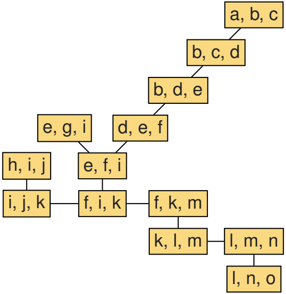




Treewidth

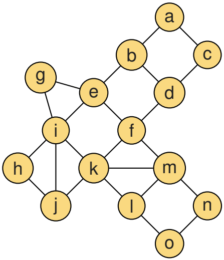
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in some bag

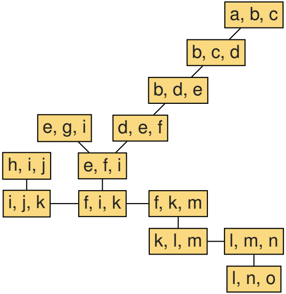




Treewidth

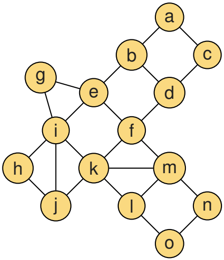
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in some bag
  2. every edge is in some bag

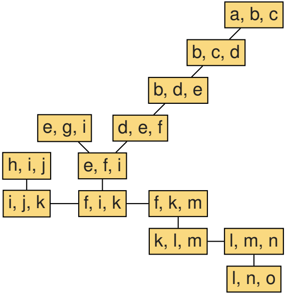




Treewidth

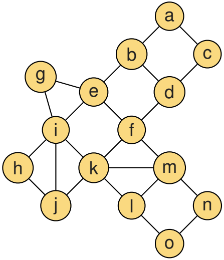
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in some bag
  2. every edge is in some bag
  3. bags containing a vertex form a connected subtree

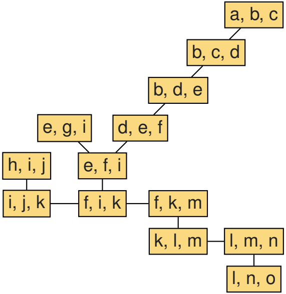




Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in some bag
  2. every edge is in some bag
  3. bags containing a vertex form a connected subtree
- Width = max bag size -1



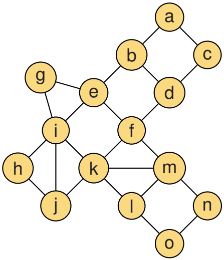


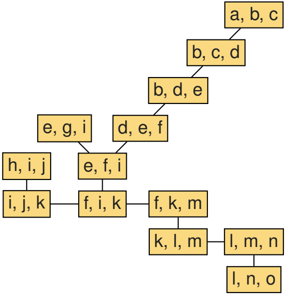
Width 2

Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in some bag
  2. every edge is in some bag
  3. bags containing a vertex form a connected subtree
- Width = max bag size -1

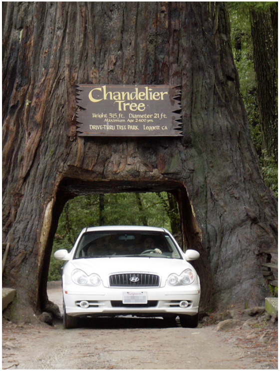
[Robertson & Seymour '84, Arnborg & Proskurowski '89, Bertele & Brioschi '72, Halin '76]





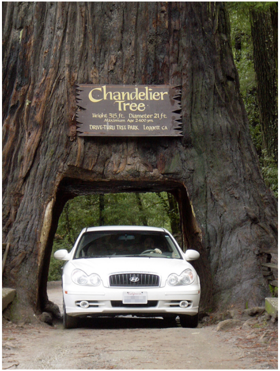
Width 2

Why treewidth



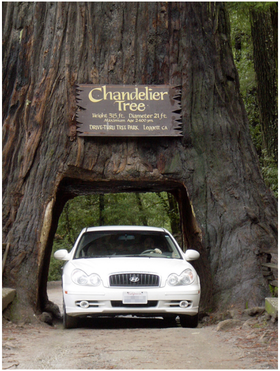
Why treewidth

- Most of NP-hard graph problems are FPT parameterized by treewidth



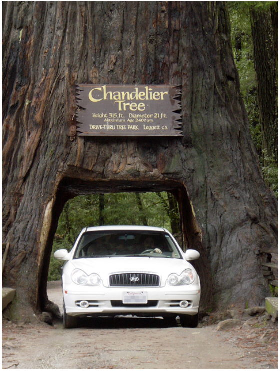
Why treewidth

- Most of NP-hard graph problems are FPT parameterized by treewidth
- Formalized by Courcelle's theorem, giving $f(k) \cdot n$ time algorithms for problems definable in $\mathbf{MSO}_2$-logic



Why treewidth

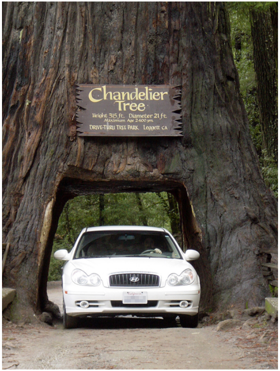
- Most of NP-hard graph problems are FPT parameterized by treewidth
- Formalized by Courcelle's theorem, giving $f(k) \cdot n$ time algorithms for problems definable in $\mathbf{MSO}_2$-logic
- Often $2^{\mathcal{O}(k)} n$ time algorithms



Why treewidth

- Most of NP-hard graph problems are FPT parameterized by treewidth
- Formalized by Courcelle's theorem, giving $f(k) \cdot n$ time algorithms for problems definable in **MSO**$_2$-logic
- Often $2^{\mathcal{O}(k)} n$ time algorithms

Need the tree decomposition!



Computing treewidth

Computing treewidth

1. Robertson-Seymour FPT-approximation and its descendants

Computing treewidth

1. Robertson-Seymour FPT-approximation and its descendants
2. Classic exact FPT algorithms

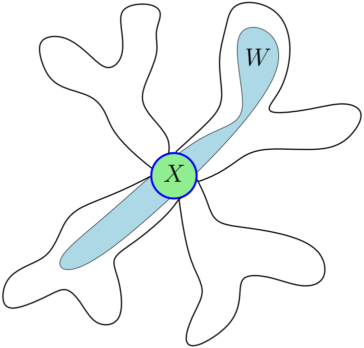
Computing treewidth

1. Robertson-Seymour FPT-approximation and its descendants
2. Classic exact FPT algorithms
3. New FPT algorithms based on local improvement

1. Robertson-Seymour FPT-approximation

Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

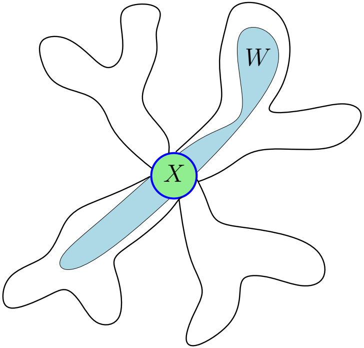


Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

Lemma

If treewidth $\leq k$, then every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$.



Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

Lemma

If treewidth $\leq k$, then every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$.

Proof:

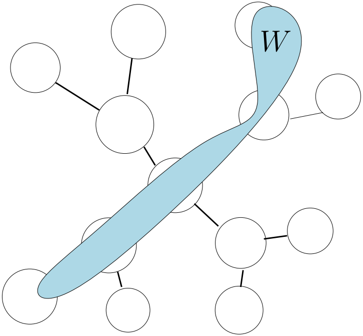
Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

Lemma

If treewidth $\leq k$, then every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$.

Proof:



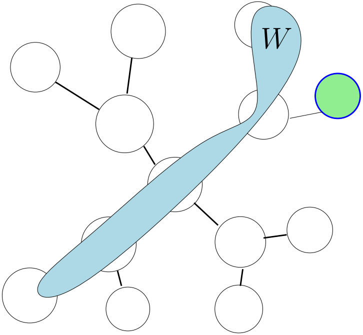
Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

Lemma

If treewidth $\leq k$, then every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$.

Proof:



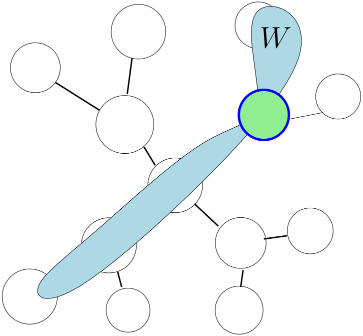
Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

Lemma

If treewidth $\leq k$, then every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$.

Proof:



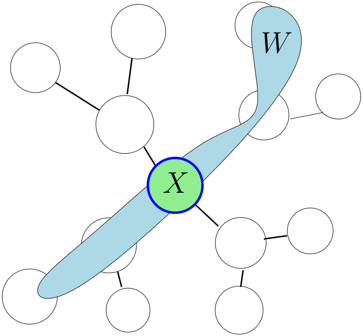
Robertson-Seymour FPT-approximation: Balanced separators

Definition: Vertex set $X \subseteq V(G)$ is a balanced separator of a vertex set $W \subseteq V(G)$ if for every component C of $G - X$ it holds that $|W \cap C| \leq |W|/2$.

Lemma

If treewidth $\leq k$, then every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$.

Proof:



Constructing the decomposition

Lemma

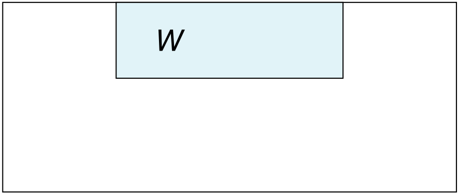
If every $W \subseteq V(G)$ has a balanced separator of size $\leq k+1$, then $\mathrm{tw}(G) \leq 4k+3$

Constructing the decomposition

Lemma

If every $W \subseteq V(G)$ has a balanced separator of size $\leq k + 1$, then $\text{tw}(G) \leq 4k + 3$

Graph

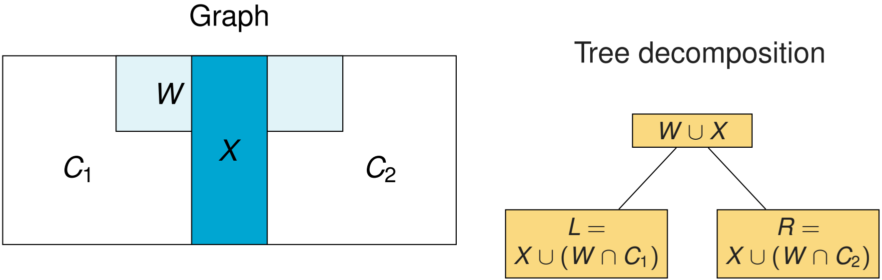


Tree decomposition

Constructing the decomposition

Lemma

If every $W \subseteq V(G)$ has a balanced separator of size $\leq k+1$, then $\text{tw}(G) \leq 4k+3$

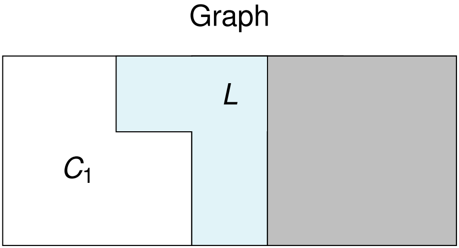


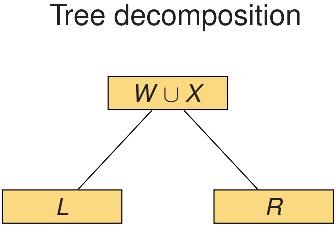
Balanced separator X with components C_1 and C_2

Constructing the decomposition

Lemma

If every $W \subseteq V(G)$ has a balanced separator of size $\leq k+1$, then $\text{tw}(G) \leq 4k+3$



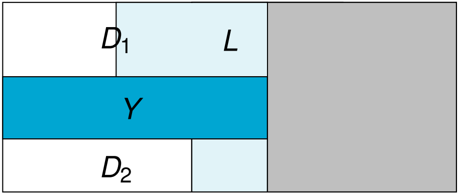


Constructing the decomposition

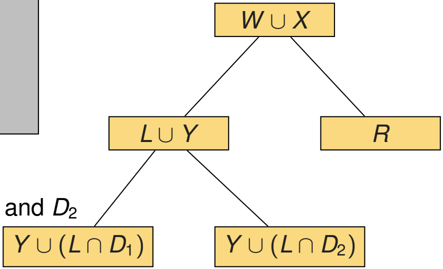
Lemma

If every $W \subseteq V(G)$ has a balanced separator of size $\leq k+1$, then $\text{tw}(G) \leq 4k+3$

Graph



Tree decomposition



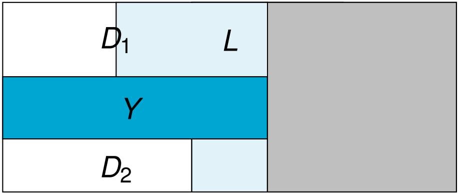
Balanced separator Y with components D_1 and D_2

Constructing the decomposition

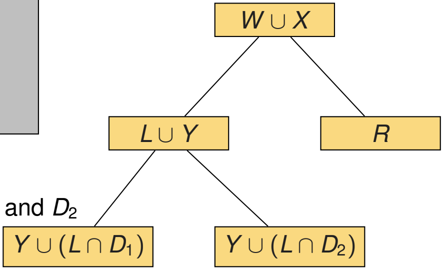
Lemma

If every $W \subseteq V(G)$ has a balanced separator of size $\leq k+1$, then $\text{tw}(G) \leq 4k+3$

Graph



Tree decomposition



Balanced separator Y with components D_1 and D_2

Continue recursively...

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)}n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$
  - [Matoušek & Thomas '91, Lagergren '91]: $\mathcal{O}(1)$-approximation in time $k^{\mathcal{O}(k)} n \log^2 n$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$
  - [Matoušek & Thomas '91, Lagergren '91]: $\mathcal{O}(1)$-approximation in time $k^{\mathcal{O}(k)} n \log^2 n$
  - [Reed '92]: 8-approximation in time $k^{\mathcal{O}(k)} n \log n$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$
  - [Matoušek & Thomas '91, Lagergren '91]: $\mathcal{O}(1)$-approximation in time $k^{\mathcal{O}(k)} n \log^2 n$
  - [Reed '92]: 8-approximation in time $k^{\mathcal{O}(k)} n \log n$
  - [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]: 3-approximation in time $2^{\mathcal{O}(k)} n \log n$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)}n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$
  - [Matoušek & Thomas '91, Lagergren '91]: $\mathcal{O}(1)$-approximation in time $k^{\mathcal{O}(k)} n \log^2 n$
  - [Reed '92]: 8-approximation in time $k^{\mathcal{O}(k)} n \log n$
  - [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]: 3-approximation in time $2^{\mathcal{O}(k)} n \log n$
    - and 5-approximation in time $2^{\mathcal{O}(k)} n$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)}n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$
  - [Matoušek & Thomas '91, Lagergren '91]: $\mathcal{O}(1)$-approximation in time $k^{\mathcal{O}(k)} n \log^2 n$
  - [Reed '92]: 8-approximation in time $k^{\mathcal{O}(k)} n \log n$
  - [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]: 3-approximation in time $2^{\mathcal{O}(k)} n \log n$
    - ⋆ and 5-approximation in time $2^{\mathcal{O}(k)} n$
  - [Fomin, Lokshtanov, Pilipczuk, Saurabh & Wrochna'18]: $\mathcal{O}(k)$-approximation in time $k^{\mathcal{O}(1)} n \log n$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- How to improve the polynomial n^2: Sometimes break the graph by a balanced separator of $V(G)$
  - [Matoušek & Thomas '91, Lagergren '91]: $\mathcal{O}(1)$-approximation in time $k^{\mathcal{O}(k)} n \log^2 n$
  - [Reed '92]: 8-approximation in time $k^{\mathcal{O}(k)} n \log n$
  - [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]: 3-approximation in time $2^{\mathcal{O}(k)} n \log n$
    - and 5-approximation in time $2^{\mathcal{O}(k)} n$
  - [Fomin, Lokshtanov, Pilipczuk, Saurabh & Wrochna'18]: $\mathcal{O}(k)$-approximation in time $k^{\mathcal{O}(1)} n \log n$
  - [Belbasi & Fürer '21]: 5-approximation in time $2^{7k} n \log n$

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- Idea applied to many other width parameters:

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- Idea applied to many other width parameters:
  - FPT-approximation of cliquewidth/rankwidth [Oum&Seymour'06], [Oum'08], matroid branchwidth [Hlinený '05], [Oum&Seymour'06], $\mathcal{H}$-treewidth [Jansen, de Kroon & Wlodarczyk '21]

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- Idea applied to many other width parameters:
  - FPT-approximation of cliquewidth/rankwidth [Oum&Seymour'06], [Oum'08], matroid branchwidth [Hlinený '05], [Oum&Seymour'06], $\mathcal{H}$-treewidth [Jansen, de Kroon & Wlodarczyk '21]
  - XP-approximation of hypertreewidth [Adler, Gottlob, Grohe '07], fractional hypertreewidth [Marx '10], and minor-matching hypertreewidth [Yolov '17]

Robertson-Seymour FPT-approximation of treewidth

Theorem (Robertson & Seymour '86)

There is a $2^{\mathcal{O}(k)} n^2$ time 4-approximation for treewidth

- Idea applied to many other width parameters:
  - FPT-approximation of cliquewidth/rankwidth [Oum&Seymour'06], [Oum'08], matroid branchwidth [Hlinený '05], [Oum&Seymour'06], $\mathcal{H}$-treewidth [Jansen, de Kroon & Wlodarczyk '21]
  - XP-approximation of hypertreewidth [Adler, Gottlob, Grohe '07], fractional hypertreewidth [Marx '10], and minor-matching hypertreewidth [Yolov '17]
  - And many more...

2. Classic exact FPT algorithms

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Proof: $\mathrm{tw}(G) \leq k$ is minor-closed

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Proof: $\text{tw}(G) \leq k$ is minor-closed

Issue: Non-uniform, non-constructive (at the time)

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Proof: $\mathrm{tw}(G) \leq k$ is minor-closed

Issue: Non-uniform, non-constructive (at the time)

- [Bodlaender & Kloks, Lagergren & Arnborg, '91]: $2^{\mathcal{O}(k^3)} n$ time dynamic programming for treewidth by Typical Sequences

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Proof: $\text{tw}(G) \leq k$ is minor-closed

Issue: Non-uniform, non-constructive (at the time)

- [Bodlaender & Kloks, Lagergren & Arnborg, '91]: $2^{\mathcal{O}(k^3)} n$ time dynamic programming for treewidth by Typical Sequences
  - Implied $2^{\mathcal{O}(k^3)} n \log^2 n$ time algorithm at the time

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Proof: $\mathrm{tw}(G) \leq k$ is minor-closed

Issue: Non-uniform, non-constructive (at the time)

- [Bodlaender & Kloks, Lagergren & Arnborg, '91]: $2^{\mathcal{O}(k^3)} n$ time dynamic programming for treewidth by Typical Sequences
  - Implied $2^{\mathcal{O}(k^3)} n \log^2 n$ time algorithm at the time
  - [Bodlaender '93]: Improvement to $2^{\mathcal{O}(k^3)} n$ by a recursive "compression" technique

Classic exact FPT algorithms

Theorem (Robertson & Seymour '86)

There is a $f(k) \cdot n^2$ time (non-uniform) algorithm for treewidth

Proof: $\text{tw}(G) \leq k$ is minor-closed

Issue: Non-uniform, non-constructive (at the time)

- [Bodlaender & Kloks, Lagergren & Arnborg, '91]: $2^{\mathcal{O}(k^3)} n$ time dynamic programming for treewidth by Typical Sequences
  - Implied $2^{\mathcal{O}(k^3)} n \log^2 n$ time algorithm at the time
  - [Bodlaender '93]: Improvement to $2^{\mathcal{O}(k^3)} n$ by a recursive "compression" technique
- Typical sequences applied to branchwidth [Bodlaender & Thilikos '97], cutwidth and carving-width [Thilikos, Serna & Bodlaender '00], rankwidth and matroid branchwidth [Jeong, Kim & Oum '18], and more...

3. New FPT algorithms based on local improvement

Local improvement based FPT 2-approximation

Local improvement based FPT 2-approximation

Theorem (K. '21)

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

Local improvement based FPT 2-approximation

Theorem (K. '21)

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

Compare to: $2^{\mathcal{O}(k)}n$ time 5-approximation of [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]

Local improvement based FPT 2-approximation

Theorem (K. '21)

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

Compare to: $2^{\mathcal{O}(k)} n$ time 5-approximation of [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]

- Breaks the 3-approximation barrier of Robertson-Seymour-type algorithms

Local improvement based FPT 2-approximation

Theorem (K. '21)

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

Compare to: $2^{\mathcal{O}(k)}n$ time 5-approximation of [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]

- Breaks the 3-approximation barrier of Robertson-Seymour-type algorithms
- Improves the $2^{\mathcal{O}(k)}$ from $\approx 2^{40k}$ to 2^{11k}

Local improvement based FPT 2-approximation

Theorem (K. '21)

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

Compare to: $2^{\mathcal{O}(k)} n$ time 5-approximation of [Bodlaender, Drange, Dregi, Fomin, Lokshtanov, & Pilipczuk '16]

- Breaks the 3-approximation barrier of Robertson-Seymour-type algorithms
- Improves the $2^{\mathcal{O}(k)}$ from $\approx 2^{40k}$ to 2^{11k}
- Techniques extended also to 2-approximating branchwidth in time $2^{\mathcal{O}(k)} n$ and rankwidth in time $2^{2^{\mathcal{O}(k)}} n^2$ [Fomin & K. '22]

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

Time complexity: $2^{\mathcal{O}(w)} n$

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2 \cdot \text{tw}(G) + 1$ then T can be improved by a certain improvement operation

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

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

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2 \cdot \mathrm{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width
   - Inspired by a proofs on Lean Tree Decompositions [Thomas '90, Bellenbaum & Diestel '02]

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

Time complexity: $2^{\mathcal{O}(w)}n$

Let T be a tree decomposition of width w

1. If $w > 2 \cdot \text{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width
   - Inspired by a proofs on Lean Tree Decompositions [Thomas '90, Bellenbaum & Diestel '02]
2. To improve the width by one, $\Omega(n)$ improvement operations may be needed

FPT 2-approximation: Outline

By the recursive compression technique of [Bodlaender '93] we can focus on:

Input: Graph G and a tree decomposition of G of width w

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

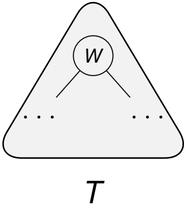
Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2 \cdot \mathrm{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width
   - Inspired by a proofs on Lean Tree Decompositions [Thomas '90, Bellenbaum & Diestel '02]
2. To improve the width by one, $\Omega(n)$ improvement operations may be needed
   - Efficient implementation by amortizatized analysis of the improvements and dynamic programming over the tree decomposition

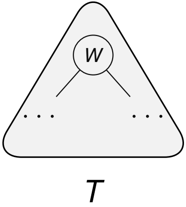
The improvement operation

- Let W be a largest bag



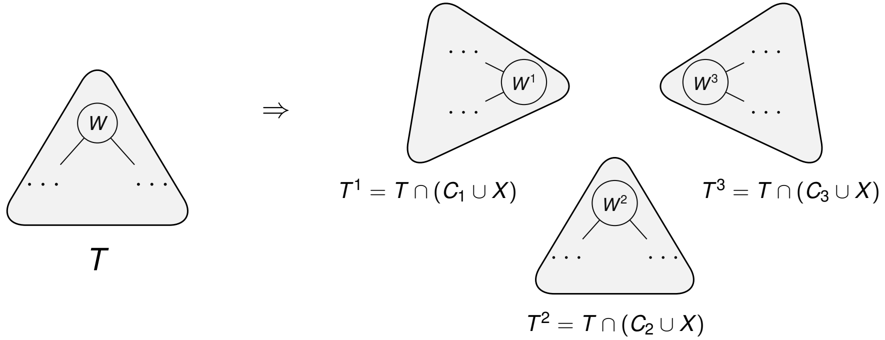
The improvement operation

- Let W be a largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of $V(G)$



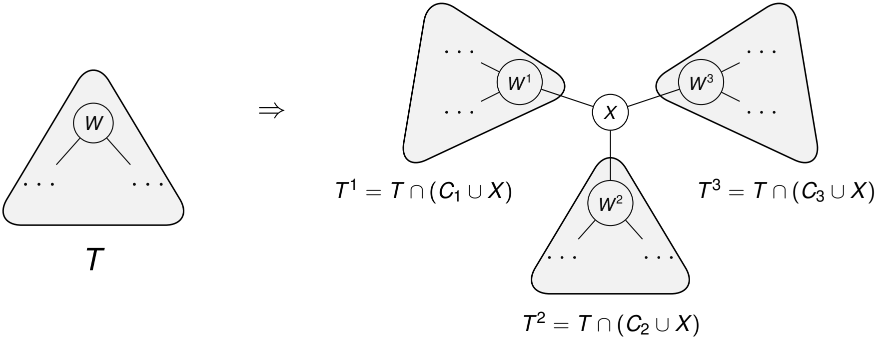
The improvement operation

- Let W be a largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of $V(G)$
- For each $i \in \{1, 2, 3\}$, 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 of T.



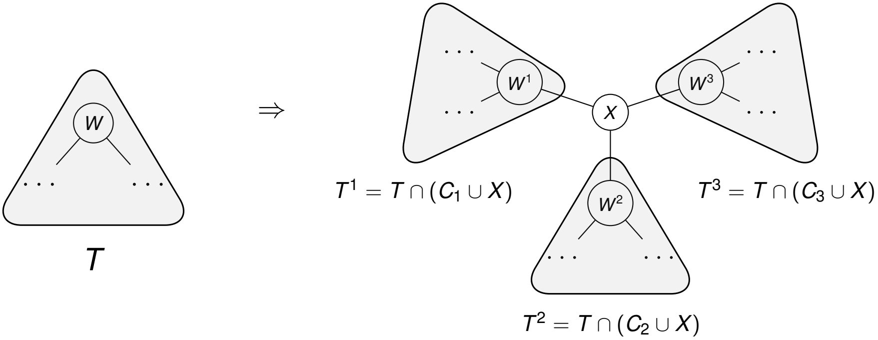
The improvement operation

- Let W be a largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of $V(G)$
- For each $i \in \{1, 2, 3\}$, 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 of T.
- The following is almost a tree decomposition of G:



The improvement operation

- Let W be a largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of $V(G)$
- For each $i \in \{1, 2, 3\}$, 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 of T.
- The following is almost a tree decomposition of G:



Except that vertices in X may violate the connectedness condition

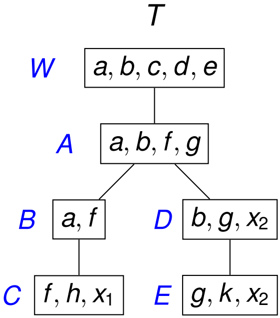
Fixing a tree decomposition

- 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:

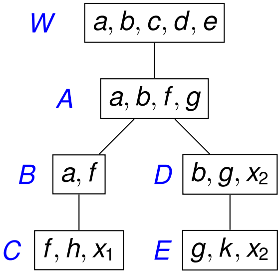


Fixing a tree decomposition

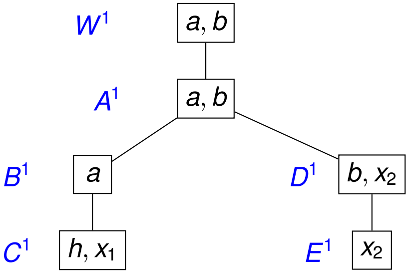
- Fix the connectedness condition by inserting vertices of X to bags

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:

T



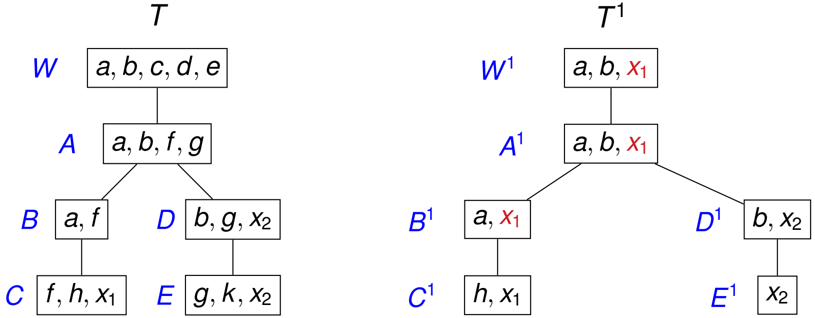
$T^1 = T \cap (C_1 \cup X)$



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:

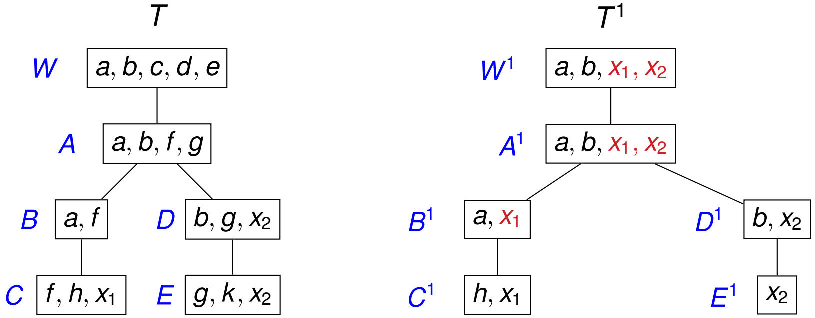


- Insert x_1 to B^1, A^1, and W^1

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

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

- $|B^i| = |B|$ holds only in a degenerate case where we can throw B^j for $j \neq i$ away

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

- $|B^i| = |B|$ holds only in a degenerate case where we can throw B^j for $j \neq i$ away
- For the bag W, $|W^i| < |W|$ is ensured by the definition of the balanced separator

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

- $|B^i| = |B|$ holds only in a degenerate case where we can throw B^j for $j \neq i$ away
- For the bag W, $|W^i| < |W|$ is ensured by the definition of the balanced separator

$\Rightarrow$ The number of bags of size $|W|$ decreases

Below 2-approximation with local improvement

Below 2-approximation with local improvement

Below 2-approximation with local improvement

- Barrier at approximation ratio 2 when using balanced separators

Below 2-approximation with local improvement

- Barrier at approximation ratio 2 when using balanced separators
- Idea: Replace balanced separator by a more general object

Below 2-approximation with local improvement

- Barrier at approximation ratio 2 when using balanced separators
- Idea: Replace balanced separator by a more general object

Theorem (K. & Lokshtanov '23)

There is a $k^{\mathcal{O}(k/\varepsilon)}n^4$ time $(1+\varepsilon)$-approximation algorithm for treewidth.

Below 2-approximation with local improvement

- Barrier at approximation ratio 2 when using balanced separators
- Idea: Replace balanced separator by a more general object

Theorem (K. & Lokshtanov '23)

There is a $k^{\mathcal{O}(k/\varepsilon)} n^4$ time $(1+\varepsilon)$-approximation algorithm for treewidth.

Theorem (K. & Lokshtanov '23)

There is a $2^{\mathcal{O}(k^2)} n^4$ time algorithm for treewidth.

Below 2-approximation with local improvement

- Barrier at approximation ratio 2 when using balanced separators
- Idea: Replace balanced separator by a more general object

Theorem (K. & Lokshtanov '23)

There is a $k^{\mathcal{O}(k/\varepsilon)} n^4$ time $(1+\varepsilon)$-approximation algorithm for treewidth.

Theorem (K. & Lokshtanov '23)

There is a $2^{\mathcal{O}(k^2)} n^4$ time algorithm for treewidth.

Asked by [Downey & Fellows '99] if the $2^{\mathcal{O}(k^3)}$ factor in Bodlaender's algorithm could be improved

Below 2-approximation with local improvement

- Barrier at approximation ratio 2 when using balanced separators
- Idea: Replace balanced separator by a more general object

Theorem (K. & Lokshtanov '23)

There is a $k^{\mathcal{O}(k/\varepsilon)} n^4$ time $(1+\varepsilon)$-approximation algorithm for treewidth.

Theorem (K. & Lokshtanov '23)

There is a $2^{\mathcal{O}(k^2)} n^4$ time algorithm for treewidth.

Asked by [Downey & Fellows '99] if the $2^{\mathcal{O}(k^3)}$ factor in Bodlaender's algorithm could be improved

- Same idea of improving a tree decomposition by decreasing the number of largest bags

Improving a tree decomposition

Let W be a largest bag of T

Improving a tree decomposition

Let W be a largest bag of T

Want to find:

Improving a tree decomposition

Let W be a largest bag of T

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and

Improving a tree decomposition

Let W be a largest bag of T

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Improving a tree decomposition

Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

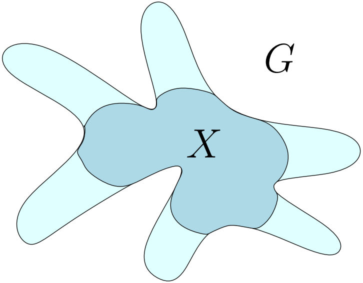
Improving a tree decomposition

Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Torso?



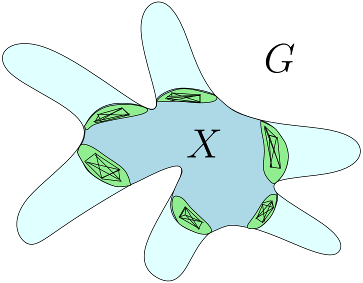
Improving a tree decomposition

Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Torso?



- Make neighborhoods of components of $G - X$ into cliques

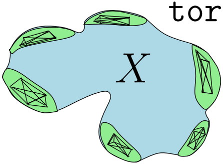
Improving a tree decomposition

Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\mathtt{torso}(X)$ of width $\leq |W| - 2$

Torso?



- Make neighborhoods of components of $G - X$ into cliques
- Delete $V(G) \setminus X$

Improving a tree decomposition

Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

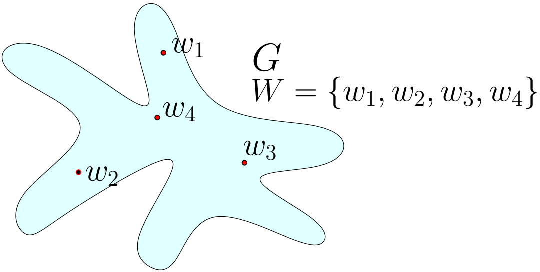
Improving a tree decomposition

Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:



Improving a tree decomposition

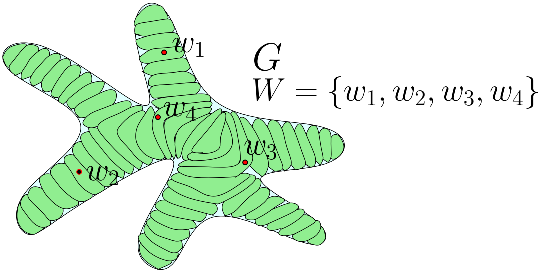
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$



Improving a tree decomposition

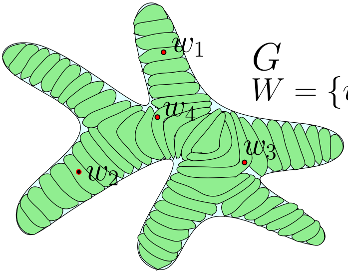
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



G

$W = \{w_1, w_2, w_3, w_4\}$

Improving a tree decomposition

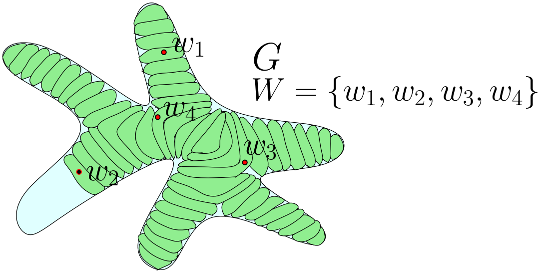
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



Improving a tree decomposition

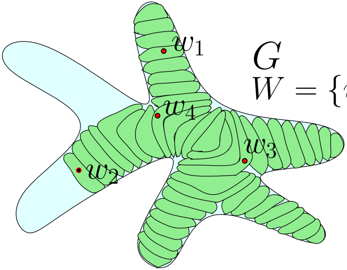
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



G
$$W = \{w_1, w_2, w_3, w_4\}$$

Improving a tree decomposition

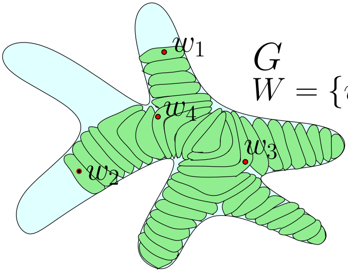
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



G
$W = \{w_1, w_2, w_3, w_4\}$

Improving a tree decomposition

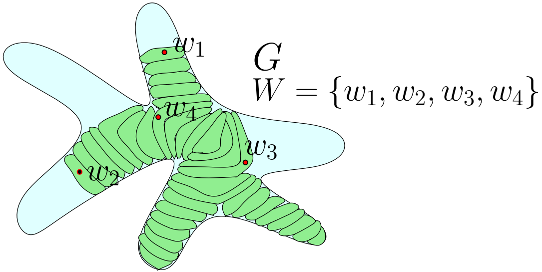
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



Improving a tree decomposition

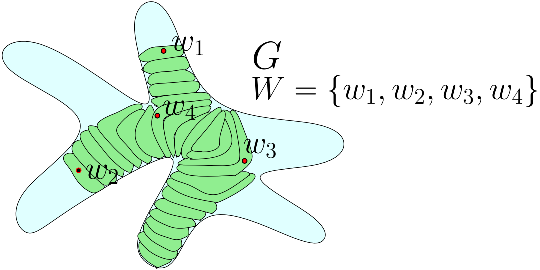
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



Improving a tree decomposition

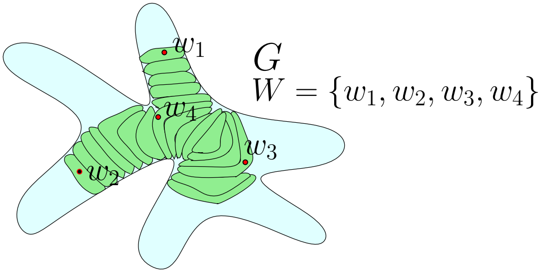
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



Improving a tree decomposition

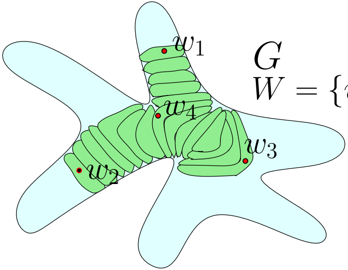
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Observations:

- If T is not optimal, then such X exists by taking $X = V(G)$
- Freedom to choose $X \subset V(G)$



G

$W = \{w_1, w_2, w_3, w_4\}$

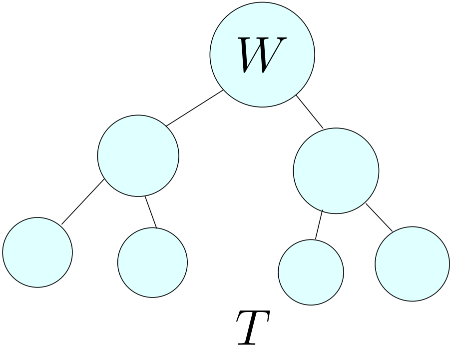
Improving a tree decomposition

Let W be a largest bag of T SUBSET TREEWIDTH

Have:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition T_X of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Improving T:



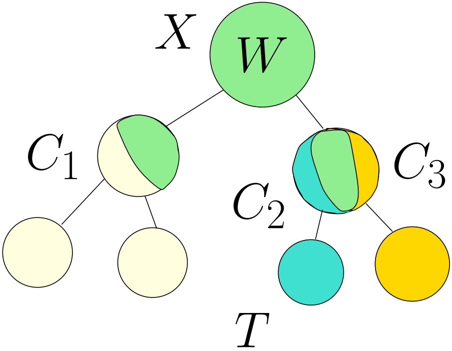
Improving a tree decomposition

Let W be a largest bag of T SUBSET TREEWIDTH

Have:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition T_X of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Improving T:



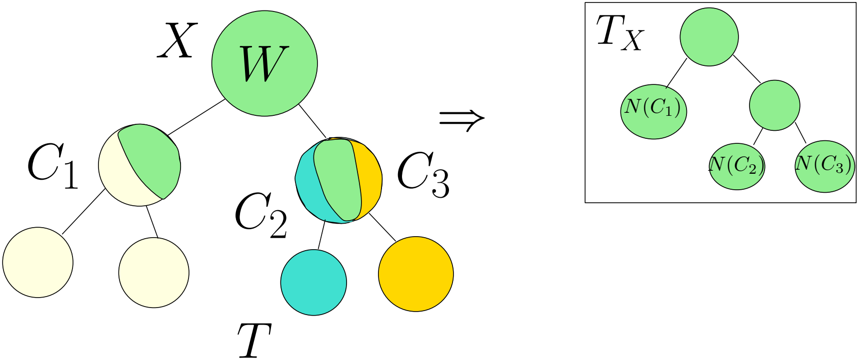
Improving a tree decomposition

Let W be a largest bag of T SUBSET TREEWIDTH

Have:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition T_X of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Improving T:



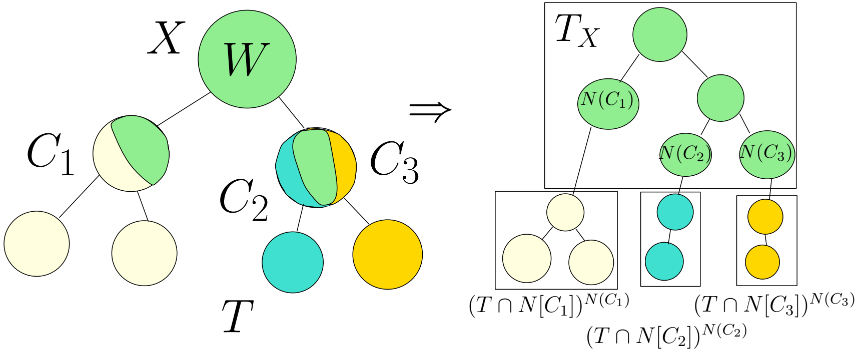
Improving a tree decomposition

Let W be a largest bag of T — SUBSET TREEWIDTH

Have:

- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition T_X of $\texttt{torso}(X)$ of width $\leq |W| - 2$

Improving T:



Subset treewidth for exact FPT algorithms

Subset treewidth for exact FPT algorithms

SUBSET TREEWIDTH

Input: Graph G, integer k, set of vertices $W \subseteq V(G)$ with $|W| = k + 2$

Output: Set $X \subseteq V(G)$ with $W \subseteq X$ and tree decomposition of $\texttt{torso}(X)$ of width $\leq k$ or that the treewidth of G is $> k$

Subset treewidth for exact FPT algorithms

SUBSET TREEWIDTH

Input: Graph G, integer k, set of vertices $W \subseteq V(G)$ with $|W| = k + 2$

Output: Set $X \subseteq V(G)$ with $W \subseteq X$ and tree decomposition of $\texttt{torso}(X)$ of width $\leq k$ or that the treewidth of G is $> k$

Theorem

If there is an $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for subset treewidth, then there is an $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for treewidth with the same function f.

Subset treewidth for exact FPT algorithms

SUBSET TREEWIDTH

Input: Graph G, integer k, set of vertices $W \subseteq V(G)$ with $|W| = k + 2$

Output: Set $X \subseteq V(G)$ with $W \subseteq X$ and tree decomposition of $\mathtt{torso}(X)$ of width $\leq k$ or that the treewidth of G is $> k$

Theorem

If there is an $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for subset treewidth, then there is an $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for treewidth with the same function f.

$2^{\mathcal{O}(k^2)} n^2$ time algorithm for subset treewidth $\rightarrow$ $2^{\mathcal{O}(k^2)} n^4$ time algorithm for treewidth

Subset treewidth for FPT-approximation

Subset treewidth for FPT-approximation

PARTITIONED SUBSET TREEWIDTH

Input: Graph G, integer k, set of vertices $W \subseteq V(G)$ with $|W| = k+2$ that is partitioned into t cliques $W_1, \ldots, W_t$

Output: Set $X \subseteq V(G)$ with $W \subseteq X$ and tree decomposition of $\texttt{torso}(X)$ of width $\leq k$ or that the treewidth of G is $> k$

Subset treewidth for FPT-approximation

PARTITIONED SUBSET TREEWIDTH

Input: Graph G, integer k, set of vertices $W \subseteq V(G)$ with $|W| = k+2$ that is partitioned into t cliques $W_1, \ldots, W_t$

Output: Set $X \subseteq V(G)$ with $W \subseteq X$ and tree decomposition of $\texttt{torso}(X)$ of width $\leq k$ or that the treewidth of G is $> k$

Theorem

If there is an $f(k,t) \cdot n^{\mathcal{O}(1)}$ time algorithm for partitioned subset treewidth, then there is a $f(\mathcal{O}(k), \mathcal{O}(1/\varepsilon)) \cdot k^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time $(1+\varepsilon)$-approximation algorithm for treewidth with the same function f.

Subset treewidth for FPT-approximation

PARTITIONED SUBSET TREEWIDTH

Input: Graph G, integer k, set of vertices $W \subseteq V(G)$ with $|W| = k+2$ that is partitioned into t cliques $W_1, \dots, W_t$

Output: Set $X \subseteq V(G)$ with $W \subseteq X$ and tree decomposition of $\texttt{torso}(X)$ of width $\leq k$ or that the treewidth of G is $> k$

Theorem

If there is an $f(k,t) \cdot n^{\mathcal{O}(1)}$ time algorithm for partitioned subset treewidth, then there is a $f(\mathcal{O}(k), \mathcal{O}(1/\varepsilon)) \cdot k^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time $(1+\varepsilon)$-approximation algorithm for treewidth with the same function f.

$k^{\mathcal{O}(kt)} n^2$ time algorithm for partitioned subset treewidth $\rightarrow$ $k^{\mathcal{O}(k/\varepsilon)} n^4$ time $(1+\varepsilon)$-approximation algorithm for treewidth

Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

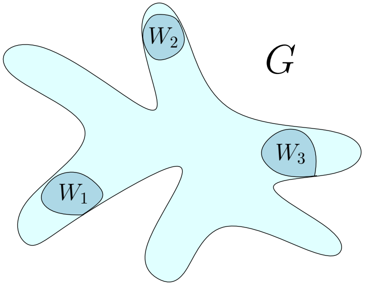
Techniques:

Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm

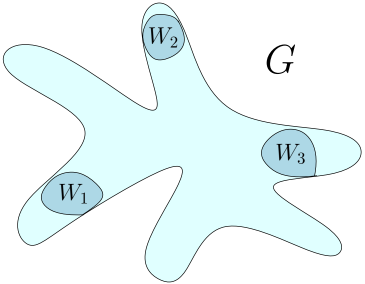


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators

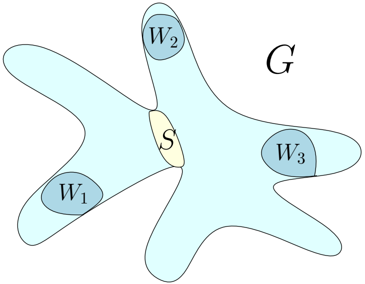


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators

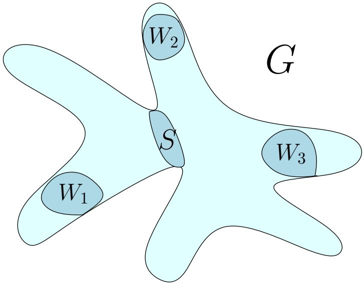


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators

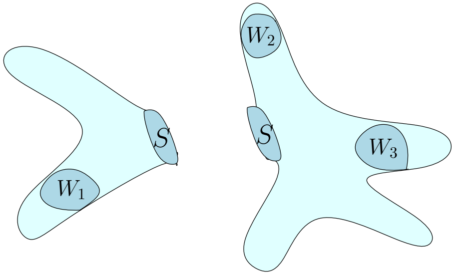


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators

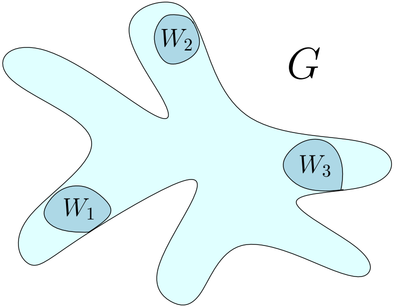


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators
- Branching on important separators [Marx '06]

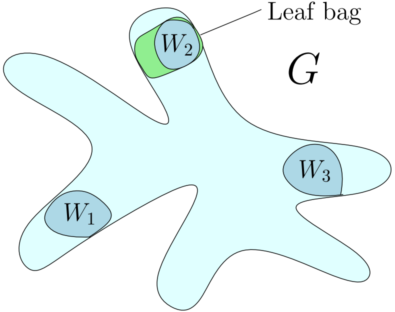


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators
- Branching on important separators [Marx '06]

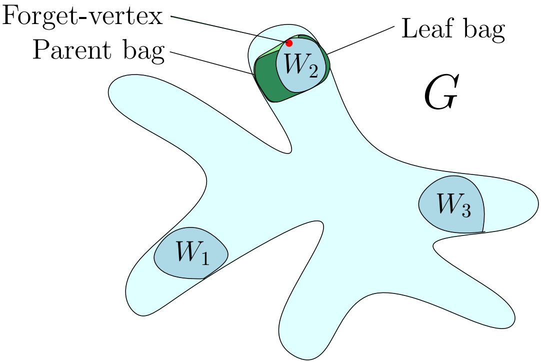


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators
- Branching on important separators [Marx '06]

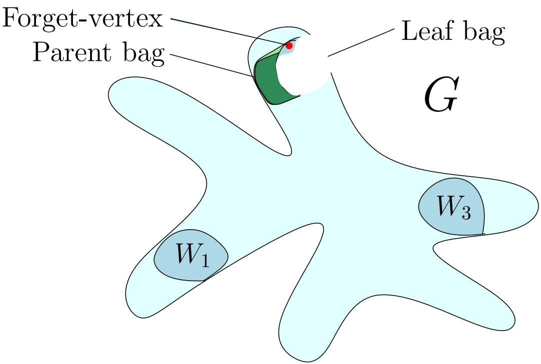


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators
- Branching on important separators [Marx '06]

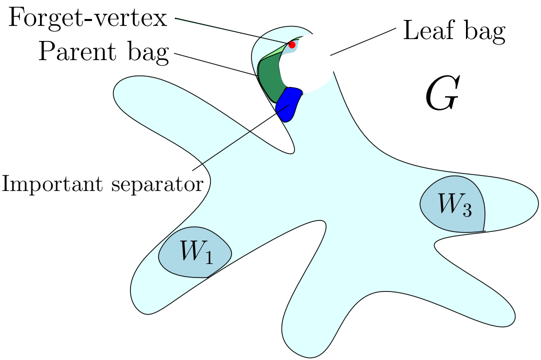


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators
- Branching on important separators [Marx '06]

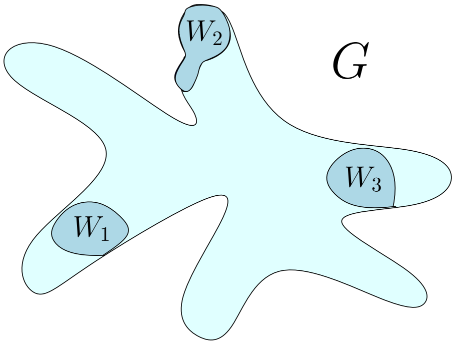


Solving Subset Treewidth

- $2^{\mathcal{O}(k^2)} n^2$ time algorithm for Subset Treewidth and $k^{\mathcal{O}(kt)} n^2$ time algorithm for Partitioned Subset Treewidth

Techniques:

- Recursive branching algorithm
- Greedy selection of safe separators
- Branching on important separators [Marx '06]



Conclusion

Classic approaches for computing width parameters:

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation
- Exact FPT via typical sequences

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation
- Exact FPT via typical sequences

New approach: Local improvement of the decomposition

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation
- Exact FPT via typical sequences

New approach: Local improvement of the decomposition

Open problems:

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation
- Exact FPT via typical sequences

New approach: Local improvement of the decomposition

Open problems:

- Prove $2^{\Omega(k)}$ lower bound for treewidth under ETH ($2^{\Omega(\sqrt{k})}$ known)

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation
- Exact FPT via typical sequences

New approach: Local improvement of the decomposition

Open problems:

- Prove $2^{\Omega(k)}$ lower bound for treewidth under ETH ($2^{\Omega(\sqrt{k})}$ known)
- Treewidth 1.9-approximation in $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time?

Conclusion

Classic approaches for computing width parameters:

- Robertson-Seymour FPT-approximation
- Exact FPT via typical sequences

New approach: Local improvement of the decomposition

Open problems:

- Prove $2^{\Omega(k)}$ lower bound for treewidth under ETH ($2^{\Omega(\sqrt{k})}$ known)
- Treewidth 1.9-approximation in $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time?
- Improve either dependence on k or n in the $2^{\mathcal{O}(k^2)} n^4$ exact treewidth algorithm

Thank you!