An Improved Parameterized Algorithm for Treewidth

Tuukka Korhonen and Daniel Lokshtanov[1]

UNIVERSITY OF BERGEN

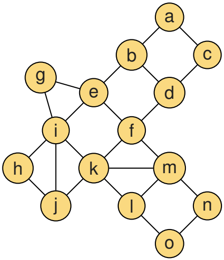
[1]University of California Santa Barbara

STOC 2023

20 June 2023

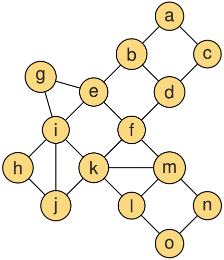
Treewidth

- Measures how close a graph is to a tree



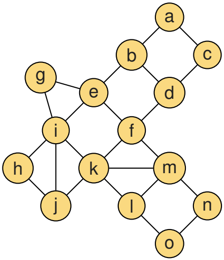
Treewidth

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



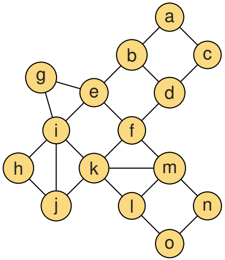
Treewidth

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



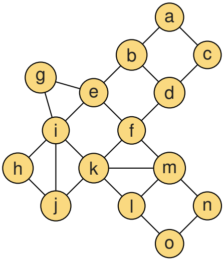
Treewidth

- 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

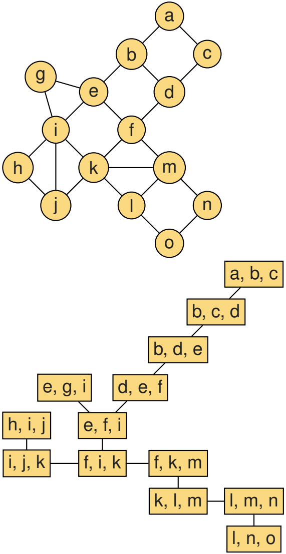
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$



Treewidth

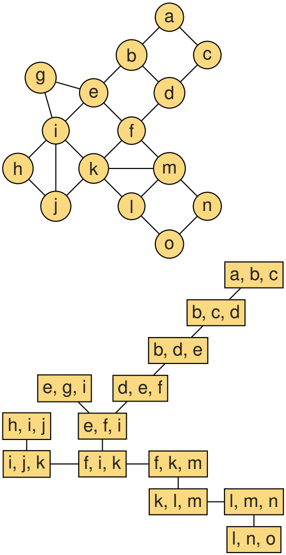
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$

- Treewidth = minimum width of a tree decomposition



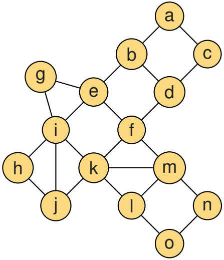
Treewidth

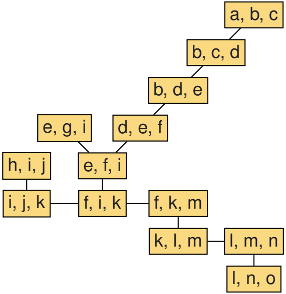
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:



Treewidth

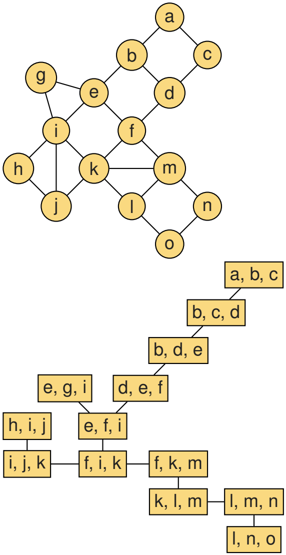
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in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





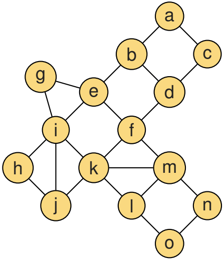
Treewidth

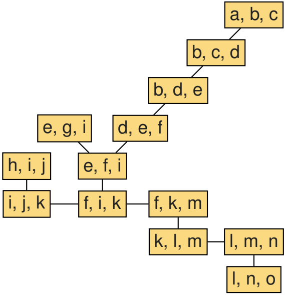
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in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



Treewidth

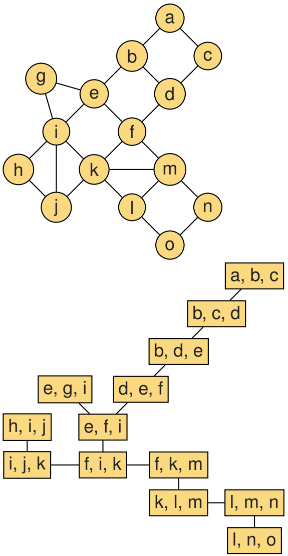
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in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

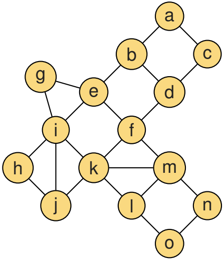
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$

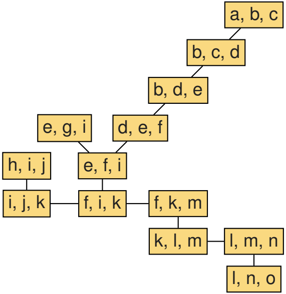
- Treewidth = minimum width of a tree decomposition

- 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

Computing treewidth: Exact FPT algorithms

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski ’87]

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time [Robertson & Seymour '86]

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time [Robertson & Seymour '86]
- $2^{\mathcal{O}(k^3)} n$ time [Bodlaender, STOC'93]

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski ’87]
- $f(k) \cdot n^2$ time [Robertson & Seymour ’86]
- $2^{\mathcal{O}(k^3)} n$ time [Bodlaender, STOC’93]
  - Using $2^{\mathcal{O}(k^3)} n$ time dynamic programming of [Bodlaender & Kloks, Lagergren & Arnborg, ’91]

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time [Robertson & Seymour '86]
- $2^{\mathcal{O}(k^3)} n$ time [Bodlaender, STOC'93]
  - Using $2^{\mathcal{O}(k^3)} n$ time dynamic programming of [Bodlaender & Kloks, Lagergren & Arnborg, '91]
- "Can the dependence $2^{\mathcal{O}(k^3)}$ on k be improved?" [Downey & Fellows '99], [Telle'06],[Bodlaender, Drange, Dregi, Fomin, Lokshtanov&Pilipczuk'16], [Bodlaender, Jaffke & Telle '20]

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski ’87]
- $f(k) \cdot n^2$ time [Robertson & Seymour ’86]
- $2^{\mathcal{O}(k^3)} n$ time [Bodlaender, STOC’93]
  - Using $2^{\mathcal{O}(k^3)} n$ time dynamic programming of [Bodlaender & Kloks, Lagergren & Arnborg, ’91]
- “Can the dependence $2^{\mathcal{O}(k^3)}$ on k be improved?” [Downey & Fellows ’99], [Telle’06],[Bodlaender, Drange, Dregi, Fomin, Lokshtanov&Pilipczuk’16], [Bodlaender, Jaffke & Telle ’20]

Theorem (This paper)

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

Computing treewidth: Exact FPT algorithms

- NP-complete [Arnborg, Corneil, Proskurowski ’87]
- $f(k) \cdot n^2$ time [Robertson & Seymour ’86]
- $2^{\mathcal{O}(k^3)} n$ time [Bodlaender, STOC’93]
  - Using $2^{\mathcal{O}(k^3)} n$ time dynamic programming of [Bodlaender & Kloks, Lagergren & Arnborg, ’91]
- “Can the dependence $2^{\mathcal{O}(k^3)}$ on k be improved?” [Downey & Fellows ’99], [Telle’06],[Bodlaender, Drange, Dregi, Fomin, Lokshtanov&Pilipczuk’16], [Bodlaender, Jaffke & Telle ’20]

Theorem (This paper)

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

- No dynamic programming, runs in space $\text{poly}(n, k)$

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]
  - Assuming SSE-conjecture, NP-hard to c-approximate for every constant c [Wu, Austrin, Pitassi & Liu '14]

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]
  - Assuming SSE-conjecture, NP-hard to c-approximate for every constant c [Wu, Austrin, Pitassi & Liu '14]
- FPT constant-approximation:

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]
  - Assuming SSE-conjecture, NP-hard to c-approximate for every constant c [Wu, Austrin, Pitassi & Liu '14]
- FPT constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]
  - Assuming SSE-conjecture, NP-hard to c-approximate for every constant c [Wu, Austrin, Pitassi & Liu '14]
- FPT constant-approximation:
  - $2^{\mathcal{O}(k)}n^2$ time 4-approximation [Robertson & Seymour '86]
  - $2^{\mathcal{O}(k)}n$ time 5-approximation [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]
  - Assuming SSE-conjecture, NP-hard to c-approximate for every constant c [Wu, Austrin, Pitassi & Liu '14]
- FPT constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]
  - $2^{\mathcal{O}(k)} n$ time 5-approximation [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $2^{\mathcal{O}(k)} n$ time 2-approximation [K. '21]

Computing treewidth: FPT-Approximation

- Polynomial-time approximation:
  - $\mathcal{O}(\sqrt{\log k})$-approximation [Feige, Hajiaghayi & Lee '08]
  - Assuming SSE-conjecture, NP-hard to c-approximate for every constant c [Wu, Austrin, Pitassi & Liu '14]
- FPT constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]
  - $2^{\mathcal{O}(k)} n$ time 5-approximation [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $2^{\mathcal{O}(k)} n$ time 2-approximation [K. '21]

Theorem (This paper)

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

Our algorithms

Overview of our algorithms

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

- Generalization of the improvement method from [K. '21]

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

- Generalization of the improvement method from [K. '21]
  - Pulling argument to re-arrange tree decompositions, originating from lean tree decompositions [Thomas '90, Bellenbaum and Diestel '02]

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

- Generalization of the improvement method from [K. '21]
  - Pulling argument to re-arrange tree decompositions, originating from lean tree decompositions [Thomas '90, Bellenbaum and Diestel '02]

2. Solving the subset treewidth problem

Theorem: $2^{\mathcal{O}(k^2)} n^2$ and $k^{\mathcal{O}(k/\varepsilon)} n^2$ time algorithms for Subset treewidth

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

- Generalization of the improvement method from [K. '21]
  - Pulling argument to re-arrange tree decompositions, originating from lean tree decompositions [Thomas '90, Bellenbaum and Diestel '02]

2. Solving the subset treewidth problem

Theorem: $2^{\mathcal{O}(k^2)}n^2$ and $k^{\mathcal{O}(k/\varepsilon)}n^2$ time algorithms for Subset treewidth

Techniques:

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

- Generalization of the improvement method from [K. '21]
  - Pulling argument to re-arrange tree decompositions, originating from lean tree decompositions [Thomas '90, Bellenbaum and Diestel '02]

2. Solving the subset treewidth problem

Theorem: $2^{\mathcal{O}(k^2)} n^2$ and $k^{\mathcal{O}(k/\varepsilon)} n^2$ time algorithms for Subset treewidth

Techniques:

- Branching on important separators [Marx '06]

Overview of our algorithms

Strategy: Iteratively improve a tree decomposition

1. How to improve a tree decomposition

Theorem: It suffices to solve the *Subset treewidth* problem

Techniques:

- Generalization of the improvement method from [K. '21]
  - Pulling argument to re-arrange tree decompositions, originating from lean tree decompositions [Thomas '90, Bellenbaum and Diestel '02]

2. Solving the subset treewidth problem

Theorem: $2^{\mathcal{O}(k^2)} n^2$ and $k^{\mathcal{O}(k/\varepsilon)} n^2$ time algorithms for Subset treewidth

Techniques:

- Branching on important separators [Marx '06]
- Together with the pulling argument

Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ SUBSET TREEWIDTH

Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ SUBSET TREEWIDTH

Want to find:

Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ SUBSET TREEWIDTH

Want to find:

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

Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k+2$ 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 $\leq k$

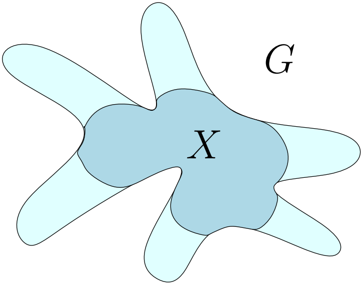
Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ — 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 $\leq k$

Torso?



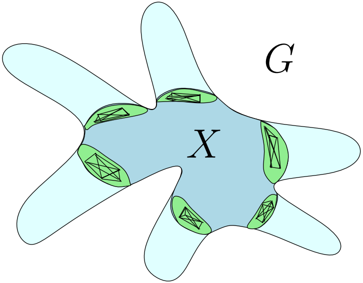
Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k+2$ — 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 $\leq k$

Torso?



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

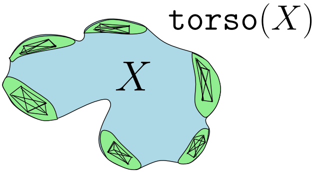
Subset treewidth

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ 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 $\leq k$

Torso?



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

Subset treewidth for exact FPT algorithms

Given graph G and $W \subseteq V(G)$ with $|W| = k+2$ — 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 $\leq k$

Subset treewidth for exact FPT algorithms

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ 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 $\leq k$

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

Given graph G and $W \subseteq V(G)$ with $|W| = k+2$ — 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 $\leq k$

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.

Proof outline:

Subset treewidth for exact FPT algorithms

Given graph G and $W \subseteq V(G)$ with $|W| = k+2$ 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 $\leq k$

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.

Proof outline:

- Solve subset treewidth where W is a largest bag of non-optimal tree decomposition T

Subset treewidth for exact FPT algorithms

Given graph G and $W \subseteq V(G)$ with $|W| = k + 2$ SUBSET TREEWIDTH

Want to find:

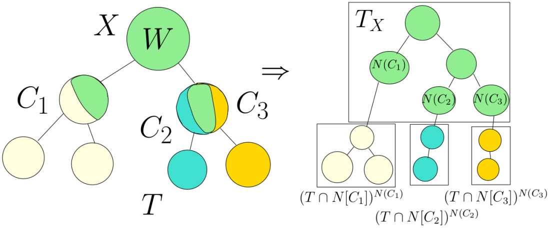
- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\mathtt{torso}(X)$ of width $\leq k$

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.

Proof outline:

- Solve subset treewidth where W is a largest bag of non-optimal tree decomposition T
- Use the output decomposition to re-arrange T



Subset treewidth for exact FPT algorithms

Given graph G and $W \subseteq V(G)$ with $|W| = k+2$ — SUBSET TREEWIDTH

Want to find:

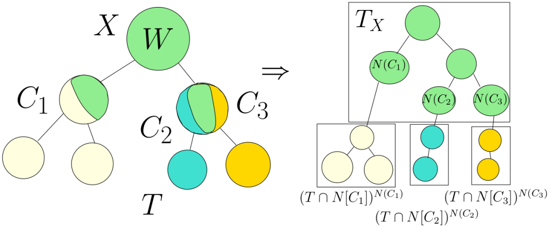
- a set X with $W \subseteq X \subseteq V(G)$, and
- a tree decomposition of $\mathtt{torso}(X)$ of width $\leq k$

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.

Proof outline:

- Solve subset treewidth where W is a largest bag of non-optimal tree decomposition T
- Use the output decomposition to re-arrange T
- Decreases the number of largest bags of T



Conclusion

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

Conclusion

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

Open questions:

Conclusion

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

Open questions:

- Prove a $2^{\Omega(k)}$ lower bound assuming ETH

Conclusion

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

Open questions:

- Prove a $2^{\Omega(k)}$ lower bound assuming ETH
  - Known reductions give $2^{\Omega(\sqrt{k})}$ lower bound

Conclusion

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

Open questions:

- Prove a $2^{\Omega(k)}$ lower bound assuming ETH
  - Known reductions give $2^{\Omega(\sqrt{k})}$ lower bound
- Treewidth 1.9-approximation in $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time?

Conclusion

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

Open questions:

- Prove a $2^{\Omega(k)}$ lower bound assuming ETH
  - Known reductions give $2^{\Omega(\sqrt{k})}$ lower bound
- Treewidth 1.9-approximation in $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time?

Thank you!