An Improved Parameterized Algorithm for Treewidth

Tuukka Korhonen and Daniel Lokshtanov[1]

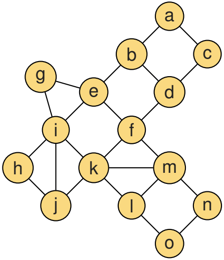
UNIVERSITY OF BERGEN

[1]University of California Santa Barbara

STOC 2023

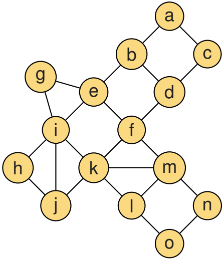
Treewidth

- Measures how close a graph is to a tree



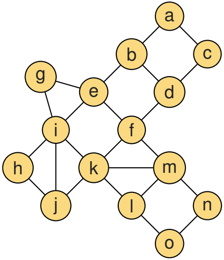
Treewidth

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



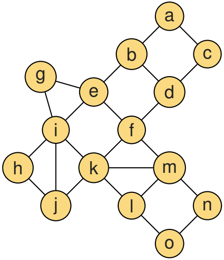
Treewidth

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



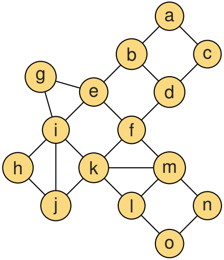
Treewidth

- 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



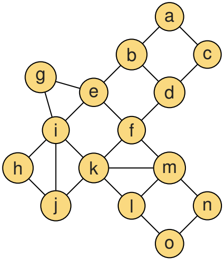
Treewidth

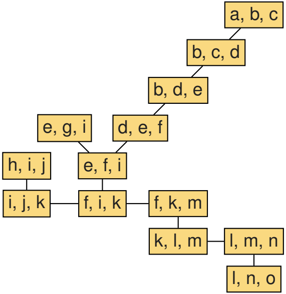
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$



Treewidth

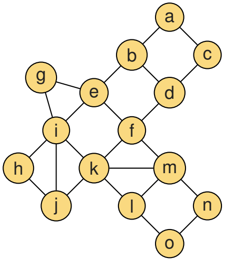
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n-1$
- Treewidth = minimum width of a tree decomposition

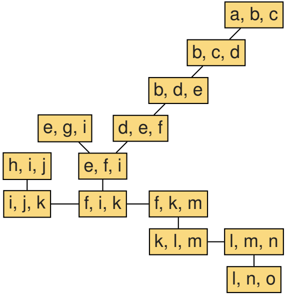




Treewidth

- 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:



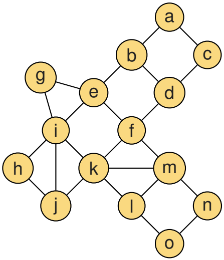


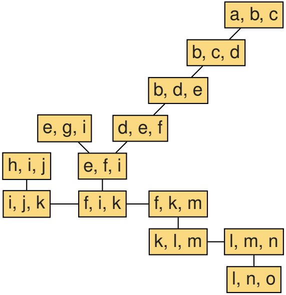
Treewidth

- 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

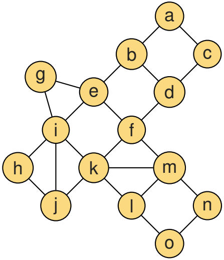
- Tree decomposition is a tree of bags so that:
  1. every vertex is in some bag

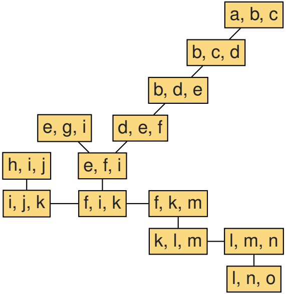




Treewidth

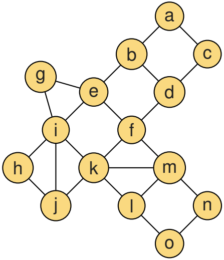
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in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

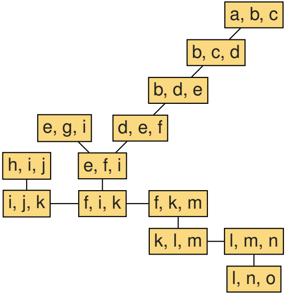




Treewidth

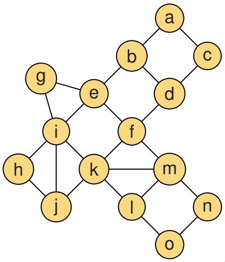
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$
- Treewidth = minimum width of a tree decomposition
- Tree decomposition is a tree of bags so that:
  1. every vertex is in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

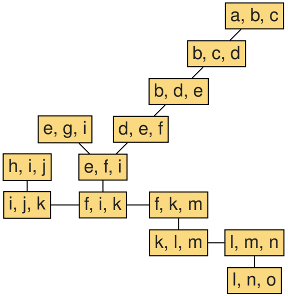




Treewidth

- 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

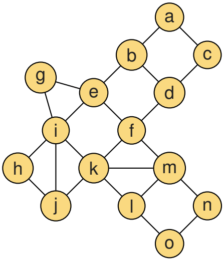
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$-grid has treewidth n
  - K_n has treewidth $n - 1$

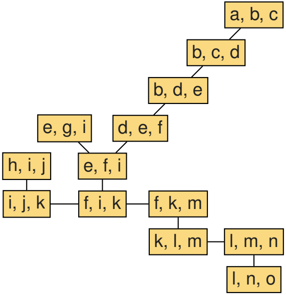
- Treewidth = minimum width of a tree decomposition

- 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

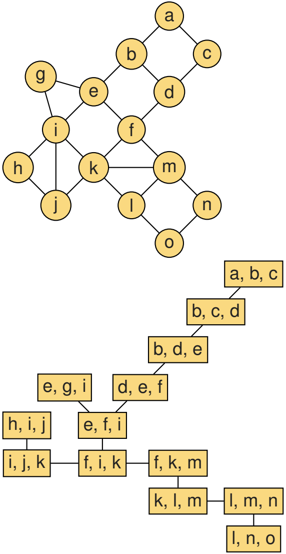
[Robertson & Seymour '84, Arnborg & Proskurowski '89, Bertele & Brioschi '72, Halin '76]





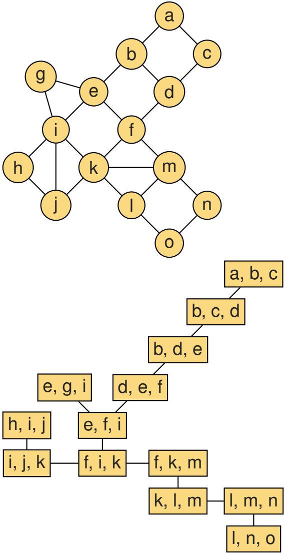
Width 2

Why treewidth: Algorithms



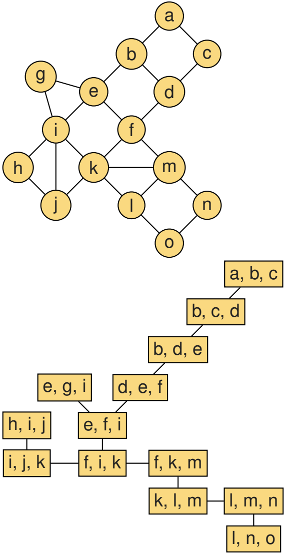
Why treewidth: Algorithms

- Algorithms on trees generalize to algorithms on graphs of small treewidth



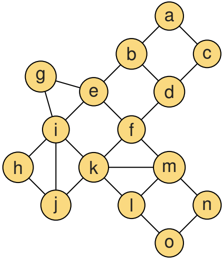
Why treewidth: Algorithms

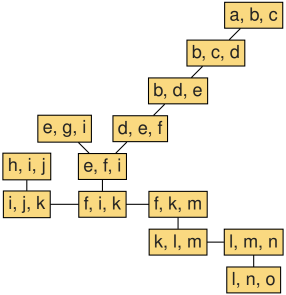
- Algorithms on trees generalize to algorithms on graphs of small treewidth
- Given a graph with a tree decomposition of width k:



Why treewidth: Algorithms

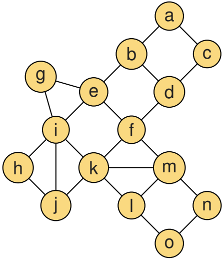
- Algorithms on trees generalize to algorithms on graphs of small treewidth
- Given a graph with a tree decomposition of width k:
- Maximum independent set in time $\mathcal{O}(2^k \cdot n)$

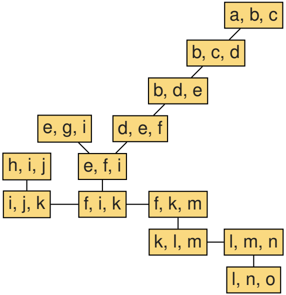




Why treewidth: Algorithms

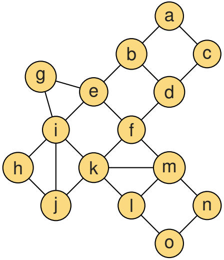
- Algorithms on trees generalize to algorithms on graphs of small treewidth
- Given a graph with a tree decomposition of width k:
- Maximum independent set in time $\mathcal{O}(2^k \cdot n)$
- Minimum dominating set in time $\mathcal{O}(3^k \cdot n)$

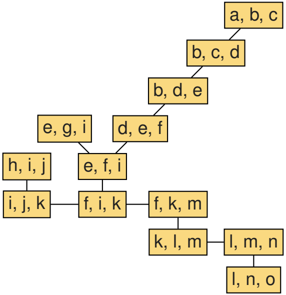




Why treewidth: Algorithms

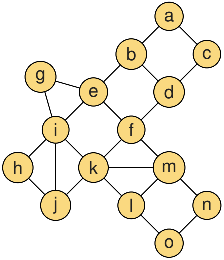
- Algorithms on trees generalize to algorithms on graphs of small treewidth
- Given a graph with a tree decomposition of width k:
- Maximum independent set in time $\mathcal{O}(2^k \cdot n)$
- Minimum dominating set in time $\mathcal{O}(3^k \cdot n)$
- Hamiltonian cycle in time $2^{\mathcal{O}(k)} \cdot n$

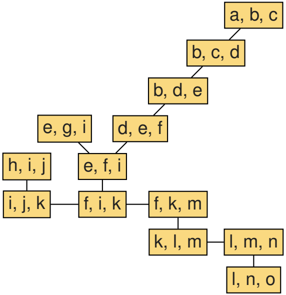




Why treewidth: Algorithms

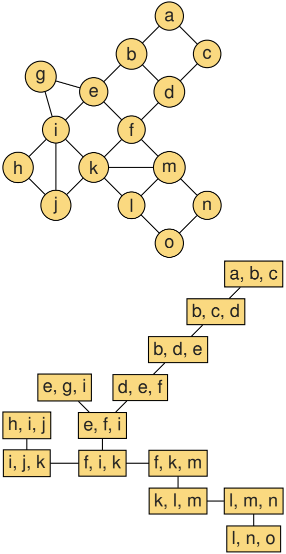
- Algorithms on trees generalize to algorithms on graphs of small treewidth
- Given a graph with a tree decomposition of width k:
- Maximum independent set in time $\mathcal{O}(2^k \cdot n)$
- Minimum dominating set in time $\mathcal{O}(3^k \cdot n)$
- Hamiltonian cycle in time $2^{\mathcal{O}(k)} \cdot n$
- Any problem in MSO-logic in time $f(k) \cdot n$





Why treewidth: Algorithms

- Algorithms on trees generalize to algorithms on graphs of small treewidth
- Given a graph with a tree decomposition of width k:
- Maximum independent set in time $\mathcal{O}(2^k \cdot n)$
- Minimum dominating set in time $\mathcal{O}(3^k \cdot n)$
- Hamiltonian cycle in time $2^{\mathcal{O}(k)} \cdot n$
- Any problem in MSO-logic in time $f(k) \cdot n$
- Need to compute the tree decomposition first!



Computing treewidth exactly

Computing treewidth exactly

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

Computing treewidth exactly

- NP-complete [Arnborg, Corneil, Proskurowski ’87]
- $\mathcal{O}(n^{k+2})$ time [Arnborg, Corneil, Proskurowski ’87]

Computing treewidth exactly

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

Computing treewidth exactly

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

Computing treewidth exactly

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $\mathcal{O}(n^{k+2})$ time [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time, non-uniform [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 exactly

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $\mathcal{O}(n^{k+2})$ time [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time, non-uniform [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]

Computing treewidth exactly

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $\mathcal{O}(n^{k+2})$ time [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time, non-uniform [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]

Theorem (This work)

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

Computing treewidth exactly

- NP-complete [Arnborg, Corneil, Proskurowski '87]
- $\mathcal{O}(n^{k+2})$ time [Arnborg, Corneil, Proskurowski '87]
- $f(k) \cdot n^2$ time, non-uniform [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]

Theorem (This work)

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

- No dynamic programming, runs in space poly(n, k)

Approximating treewidth

- Polynomial-time approximation:

Approximating treewidth

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

Approximating treewidth

- 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]

Approximating treewidth

- 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]
- $f(k) \cdot \text{poly}(n)$-time constant-approximation:

Approximating treewidth

- 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]
- $f(k) \cdot \text{poly}(n)$-time constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]

Approximating treewidth

- 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]
- $f(k) \cdot \text{poly}(n)$-time constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]
  - $k^{\mathcal{O}(k)} n \log^2 n$ [Matoušek and Thomas '91, Lagergren '91] and $k^{\mathcal{O}(k)} n \log n$ time [Reed '92] approximations

Approximating treewidth

- 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]
- $f(k) \cdot \text{poly}(n)$-time constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]
  - $k^{\mathcal{O}(k)} n \log^2 n$ [Matoušek and Thomas '91, Lagergren '91] and $k^{\mathcal{O}(k)} n \log n$ time [Reed '92] approximations
  - $2^{\mathcal{O}(k)} n$ time 5-approximation [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]

Approximating treewidth

- 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]
- $f(k) \cdot \text{poly}(n)$-time constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]
  - $k^{\mathcal{O}(k)} n \log^2 n$ [Matoušek and Thomas '91, Lagergren '91] and $k^{\mathcal{O}(k)} n \log n$ time [Reed '92] approximations
  - $2^{\mathcal{O}(k)} n$ time 5-approximation [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $2^{\mathcal{O}(k)} n$ time 2-approximation [K. '21]

Approximating treewidth

- 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]
- $f(k) \cdot \text{poly}(n)$-time constant-approximation:
  - $2^{\mathcal{O}(k)} n^2$ time 4-approximation [Robertson & Seymour '86]
  - $k^{\mathcal{O}(k)} n \log^2 n$ [Matoušek and Thomas '91, Lagergren '91] and $k^{\mathcal{O}(k)} n \log n$ time [Reed '92] approximations
  - $2^{\mathcal{O}(k)} n$ time 5-approximation [Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '16]
  - $2^{\mathcal{O}(k)} n$ time 2-approximation [K. '21]

Theorem (This work)

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

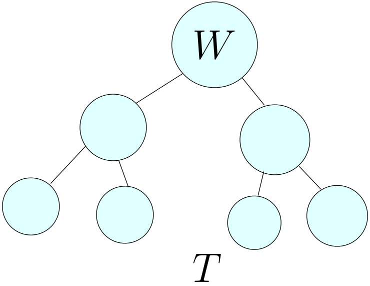
Techniques:

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

Improving a tree decomposition

Setting

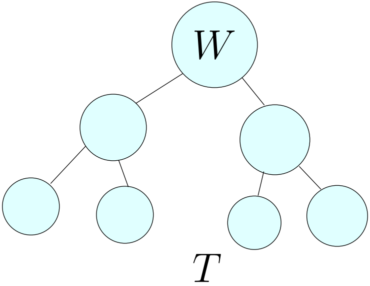
Suppose we have a tree decomposition T whose largest bag is W



Setting

Suppose we have a tree decomposition T whose largest bag is W

Goal:

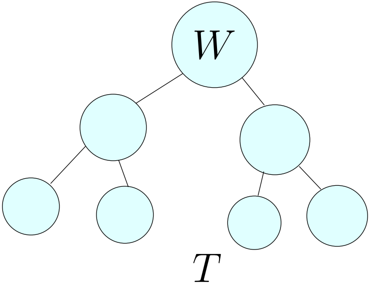


Setting

Suppose we have a tree decomposition T whose largest bag is W

Goal:

1. either decrease the number of bags of size $|W|$ while not increasing the width of T, or

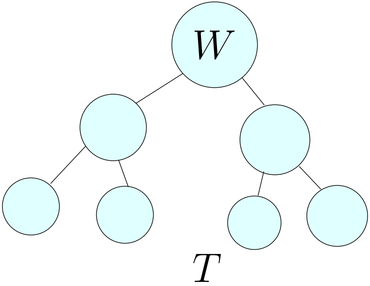


Setting

Suppose we have a tree decomposition T whose largest bag is W

Goal:

1. either decrease the number of bags of size $|W|$ while not increasing the width of T, or
2. conclude that T is (approximately) optimal



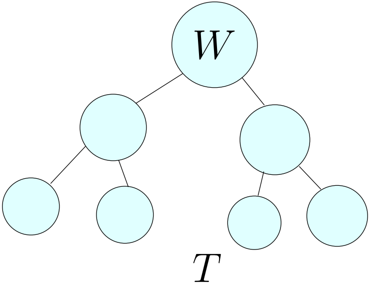
Setting

Suppose we have a tree decomposition T whose largest bag is W

Goal:

1. either decrease the number of bags of size $|W|$ while not increasing the width of T, or
2. conclude that T is (approximately) optimal

Repeat for $\mathcal{O}(\text{tw}(G) \cdot n)$ iterations to get an (approximately) optimal tree decomposition



Setting

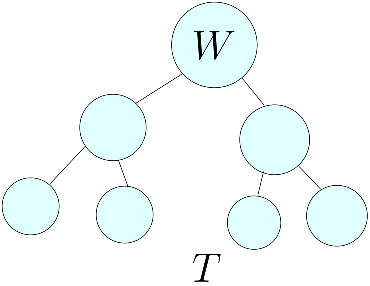
Suppose we have a tree decomposition T whose largest bag is W

Goal:

1. either decrease the number of bags of size $|W|$ while not increasing the width of T, or
2. conclude that T is (approximately) optimal

Repeat for $\mathcal{O}(\text{tw}(G) \cdot n)$ iterations to get an (approximately) optimal tree decomposition

(assume to start with width $\mathcal{O}(\text{tw}(G))$ decomposition)



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 $\mathtt{torso}(X)$ of width $\leq |W| - 2$

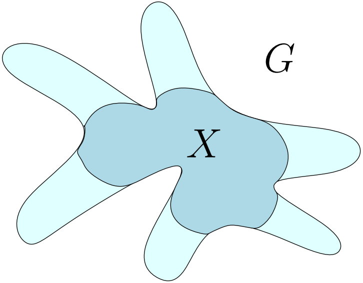
Improving a tree decomposition

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?



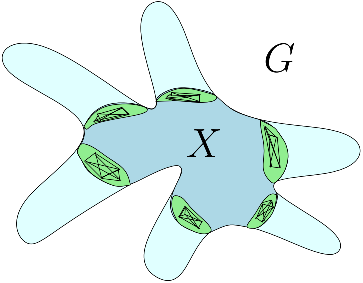
Improving a tree decomposition

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 $G \setminus X$ into cliques

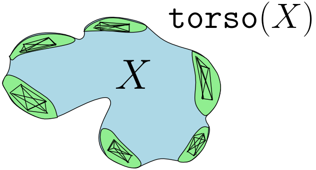
Improving a tree decomposition

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 $G \setminus X$ 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:

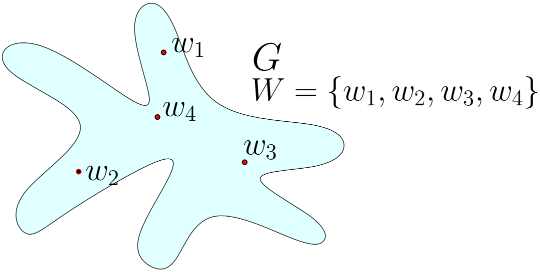
Improving a tree decomposition

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 $\mathtt{torso}(X)$ of width $\leq |W| - 2$

Observations:



Improving a tree decomposition

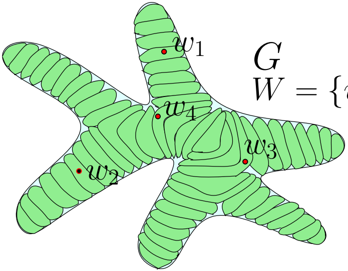
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

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

Observations:

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



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

Improving a tree decomposition

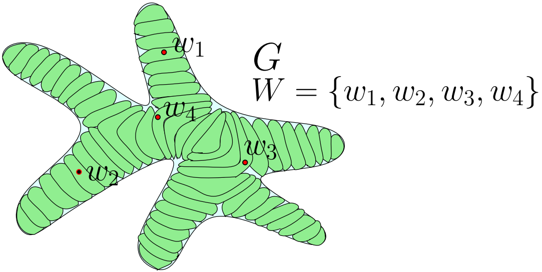
Let W be a largest bag of T — SUBSET TREEWIDTH

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

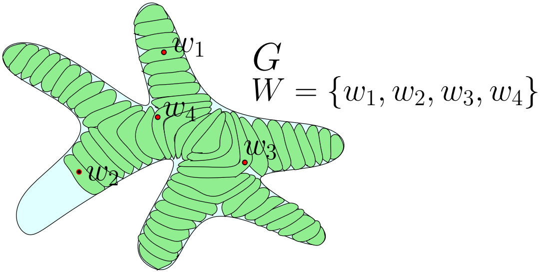
Let W be a largest bag of T — SUBSET TREEWIDTH

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

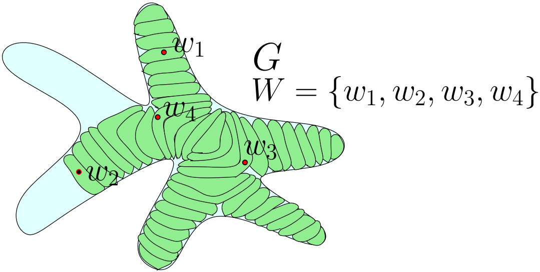
Let W be a largest bag of T — SUBSET TREEWIDTH

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

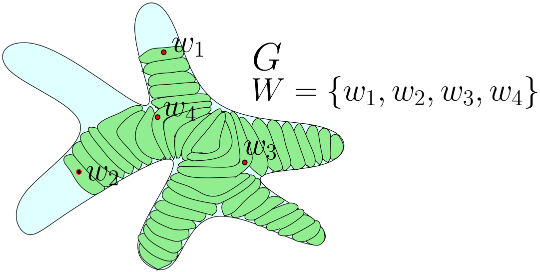
Let W be a largest bag of T — SUBSET TREEWIDTH

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 $X \subset V(G)$



Improving a tree decomposition

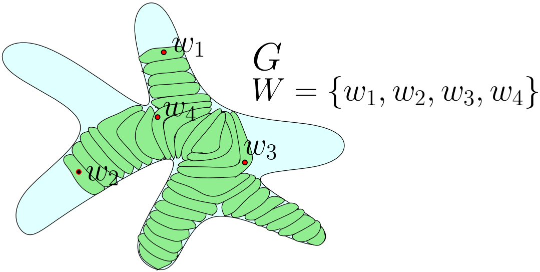
Let W be a largest bag of T — SUBSET TREEWIDTH

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

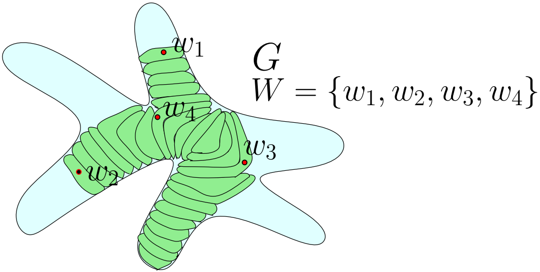
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

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

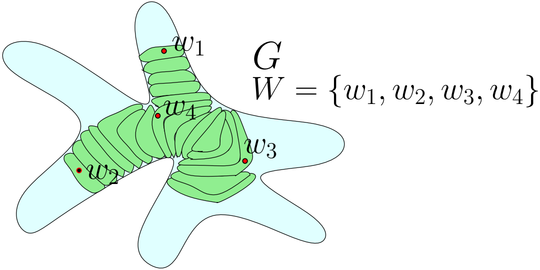
Let W be a largest bag of T — SUBSET TREEWIDTH

Want to find:

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

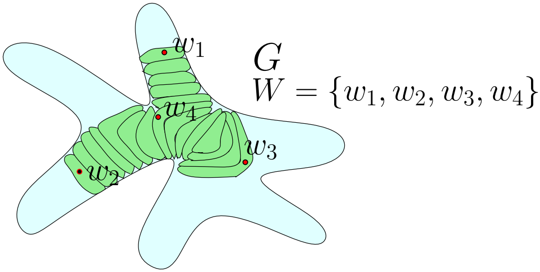
Let W be a largest bag of T — SUBSET TREEWIDTH

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 $X \subset V(G)$



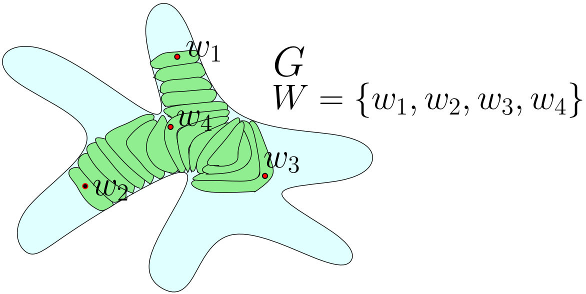
Improving a tree decomposition

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 $\leq |W| - 2$

Big-leaf formulation:



Improving a tree decomposition

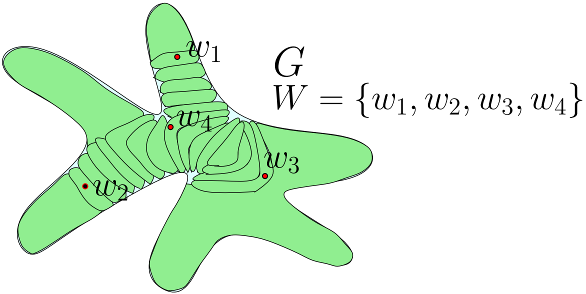
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

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

Big-leaf formulation:

- Find a tree decomposition of G whose internal bags have size $\leq |W| - 1$ and cover W, but leaf bags can be arbitrarily large



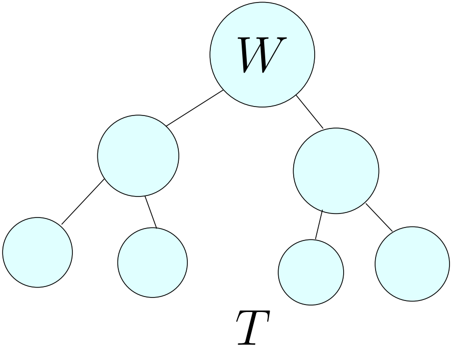
Improving a tree decomposition

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:



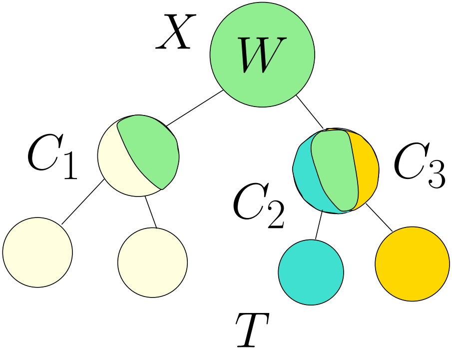
Improving a tree decomposition

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:



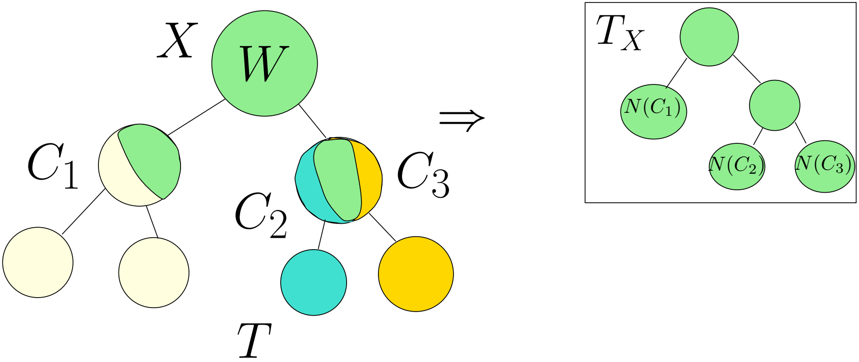
Improving a tree decomposition

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:



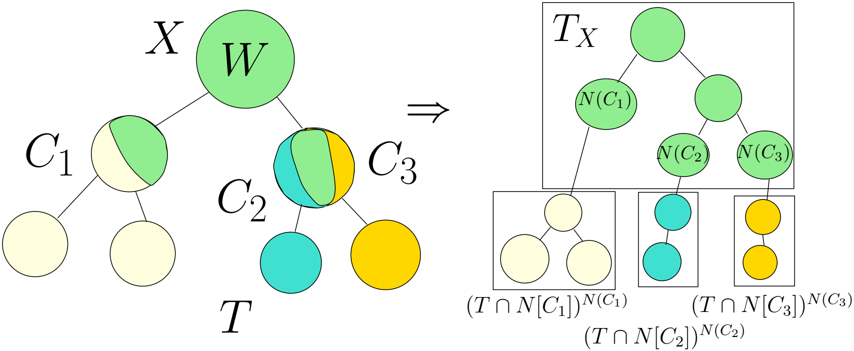
Improving a tree decomposition

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 algorithms

Subset treewidth for exact 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 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 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.

(actually *if and only if*)

Subset treewidth for exact 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.

(actually *if and only if*)

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

Subset treewidth for approximation schemes

Subset treewidth for approximation schemes

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 $> k$

Subset treewidth for approximation schemes

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 approximation schemes

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.

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

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
- The first improvement on $f(k)$ in $f(k) \cdot \text{poly}(n)$ treewidth algorithms since [Bodlaender & Kloks, Lagergren & Arnborg, '91]

Conclusion

- $2^{\mathcal{O}(k^2)}n^4$ time algorithm and $k^{\mathcal{O}(k/\varepsilon)}n^4$ time $(1+\varepsilon)$-approximation for treewidth
- The first improvement on $f(k)$ in $f(k) \cdot \text{poly}(n)$ treewidth algorithms since [Bodlaender & Kloks, Lagergren & Arnborg, '91]

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
- The first improvement on $f(k)$ in $f(k) \cdot \mathrm{poly}(n)$ treewidth algorithms since [Bodlaender & Kloks, Lagergren & Arnborg, '91]

Open questions:

- What is the best $f(k)$ so that treewidth can be computed in $f(k) \cdot \mathrm{poly}(n)$ 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
- The first improvement on $f(k)$ in $f(k) \cdot \mathrm{poly}(n)$ treewidth algorithms since [Bodlaender & Kloks, Lagergren & Arnborg, '91]

Open questions:

- What is the best $f(k)$ so that treewidth can be computed in $f(k) \cdot \mathrm{poly}(n)$ time?
- 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
- The first improvement on $f(k)$ in $f(k) \cdot \mathrm{poly}(n)$ treewidth algorithms since [Bodlaender & Kloks, Lagergren & Arnborg, '91]

Open questions:

- What is the best $f(k)$ so that treewidth can be computed in $f(k) \cdot \mathrm{poly}(n)$ time?
- Prove a $2^{\Omega(k)}$ lower bound assuming ETH
  - Known reductions give $2^{\Omega(\sqrt{k})}$ lower bound

Thank you!

- Longer talk: https://www.youtube.com/watch?v=9If417oeWcQ
- Slides: https://tuukkakorhonen.com/