FPT Algorithms for Treewidth via Local Improvement

Tuukka Korhonen

UNIVERSITY OF BERGEN

based on joint work with
Daniel Lokshtanov[1]

[1] University of California Santa Barbara

Graph Decompositions: Small Width, Big Challenges

27 October 2022

New Results on Computing Treewidth

New Results on Computing Treewidth

Theorem (K. '21)

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

New Results on Computing Treewidth

Theorem (K. '21)

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

[Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '13]: $2^{\mathcal{O}(k)}n$ time 5-approximation

New Results on Computing Treewidth

Theorem (K. '21)

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

[Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '13]: $2^{\mathcal{O}(k)}n$ time 5-approximation

Theorem (K. & Lokshtanov '22+)

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

New Results on Computing Treewidth

Theorem (K. '21)

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

[Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '13]: $2^{\mathcal{O}(k)}n$ time 5-approximation

Theorem (K. & Lokshtanov '22+)

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

[Bodlaender '93]: $2^{\mathcal{O}(k^3)}n$ time

New Results on Computing Treewidth

Theorem (K. '21)

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

[Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '13]: $2^{\mathcal{O}(k)}n$ time 5-approximation

Theorem (K. & Lokshtanov '22+)

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

[Bodlaender '93]: $2^{\mathcal{O}(k^3)}n$ time

(based on $2^{\mathcal{O}(k^3)}n$ time dynamic programming of [Bodlaender & Kloks '91])

New Results on Computing Treewidth

Theorem (K. '21)

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

[Bodlaender, Drange, Dregi, Fomin, Lokshtanov & Pilipczuk '13]: $2^{\mathcal{O}(k)}n$ time 5-approximation

Theorem (K. & Lokshtanov '22+)

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

[Bodlaender '93]: $2^{\mathcal{O}(k^3)}n$ time

(based on $2^{\mathcal{O}(k^3)}n$ time dynamic programming of [Bodlaender & Kloks '91])

Theorem (K. & Lokshtanov '22+)

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

Outline

Outline

1. How to improve a tree decomposition

Suffices to solve the *Subset treewidth* problem

Outline

1. How to improve a tree decomposition

Suffices to solve the *Subset treewidth* problem

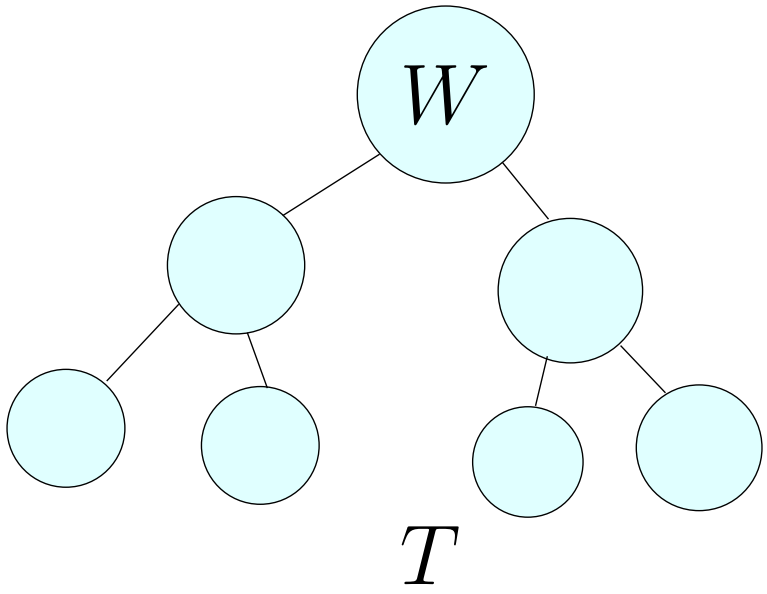
2. Solving the subset treewidth problem

Algorithms for subset treewidth that then imply algorithms for treewidth

How to improve a tree decomposition

Setting

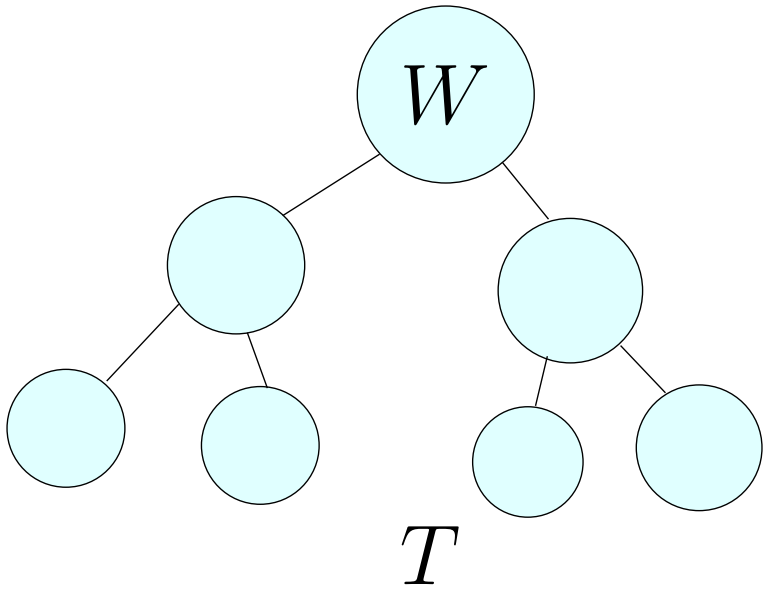
Suppose we have a tree decomposition T whose largest bag is W



Setting

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

Goal:

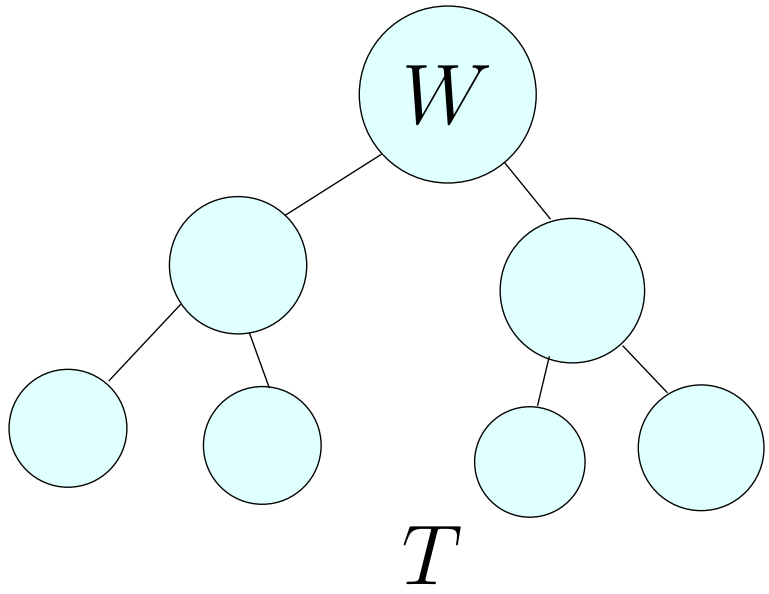


Setting

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

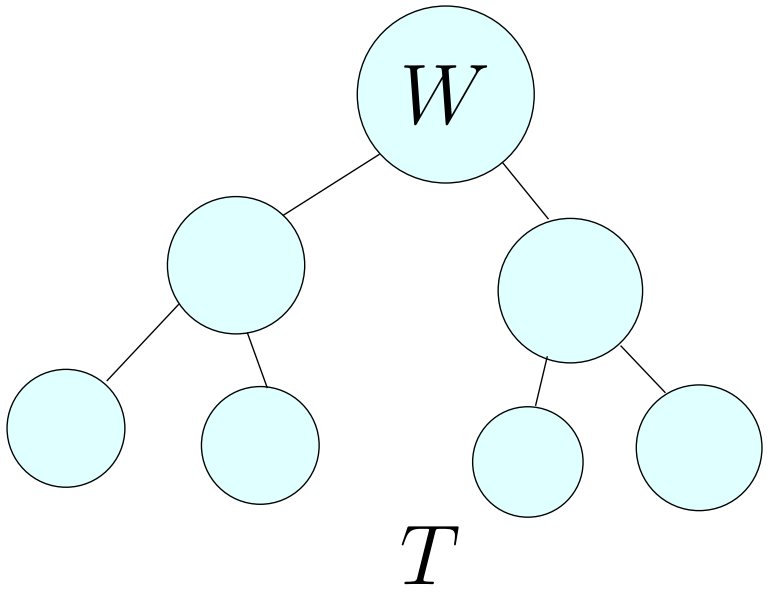


Setting

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



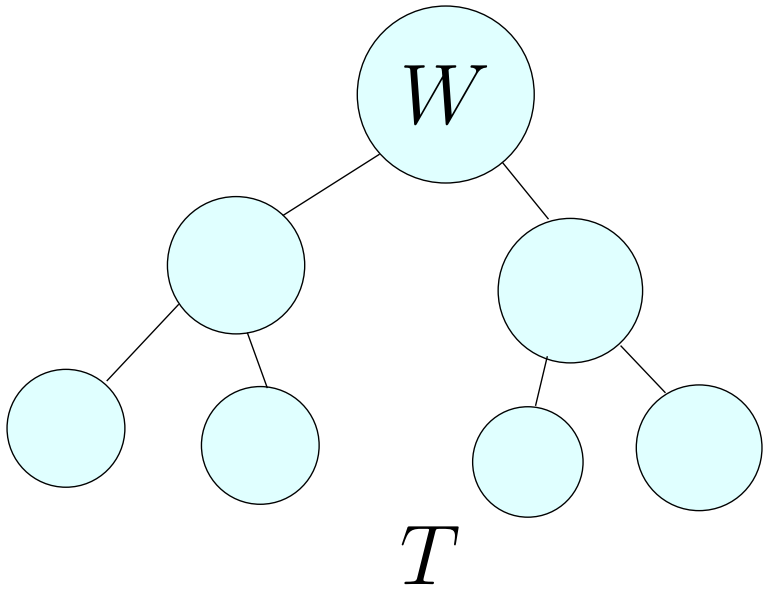
Setting

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



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 $\leq |W| - 2$

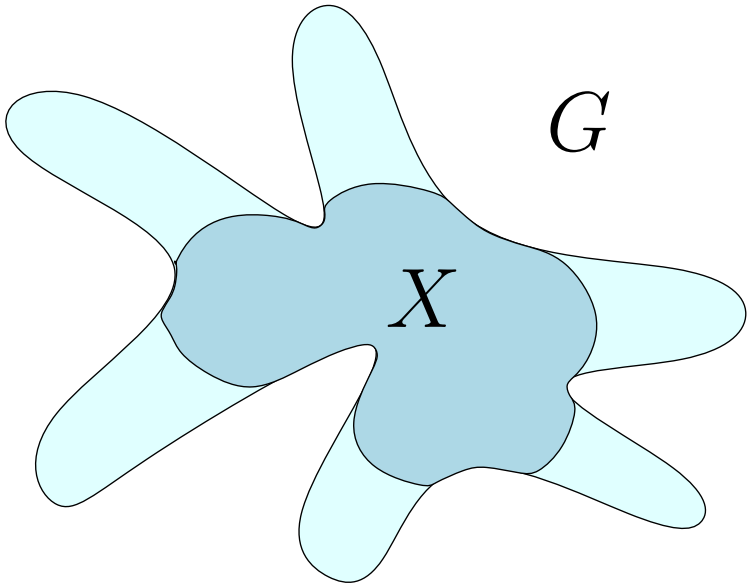
Improving a tree decomposition

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?



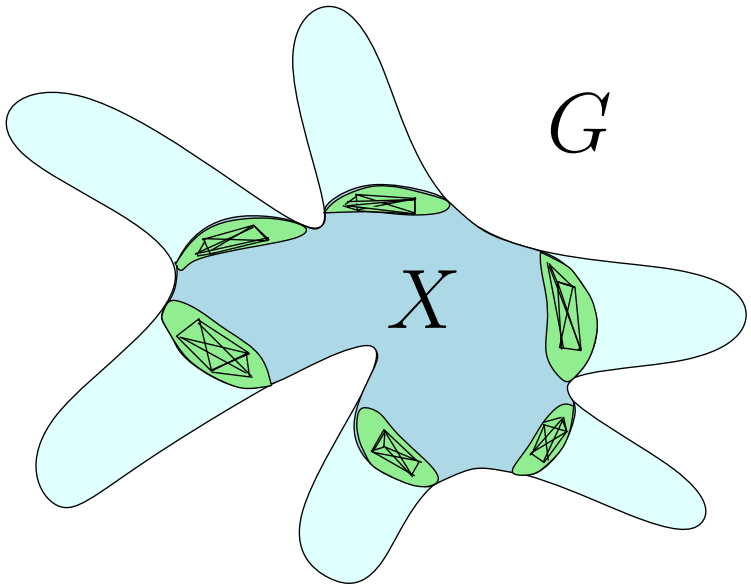
Improving a tree decomposition

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 $G \setminus X$ into cliques

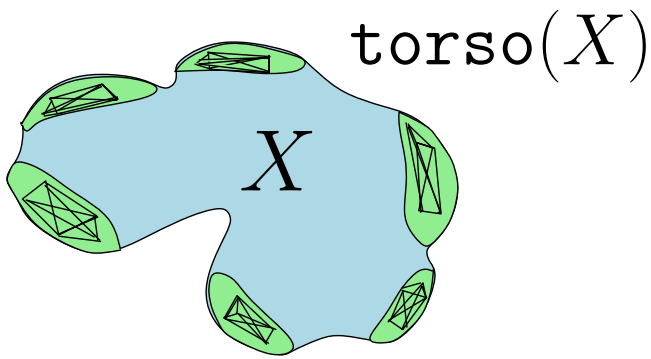
Improving a tree decomposition

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 $\texttt{torso}(X)$ 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:

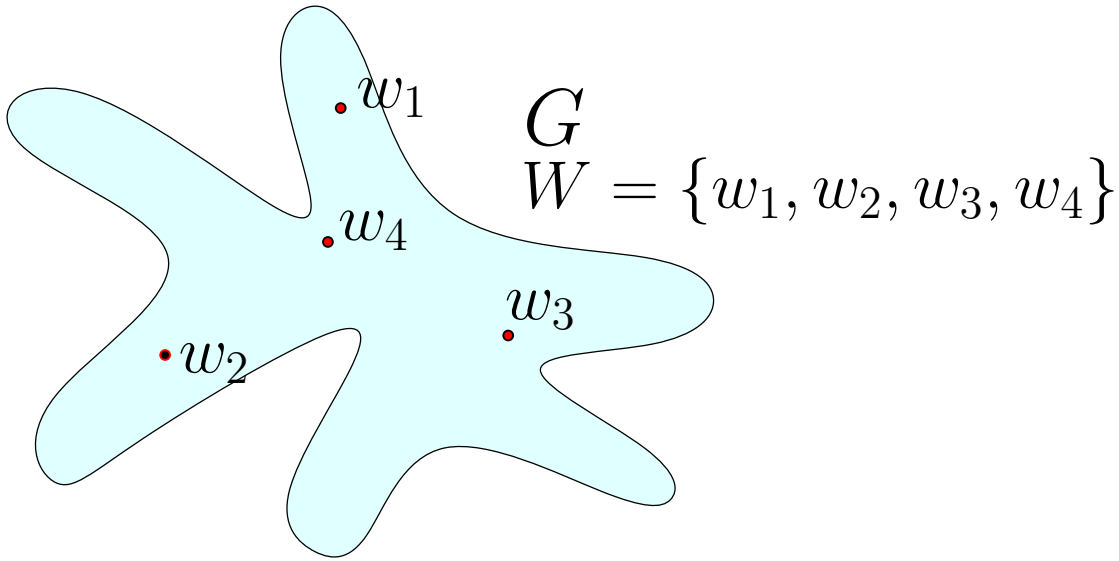
Improving a tree decomposition

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

Observations:



Improving a tree decomposition

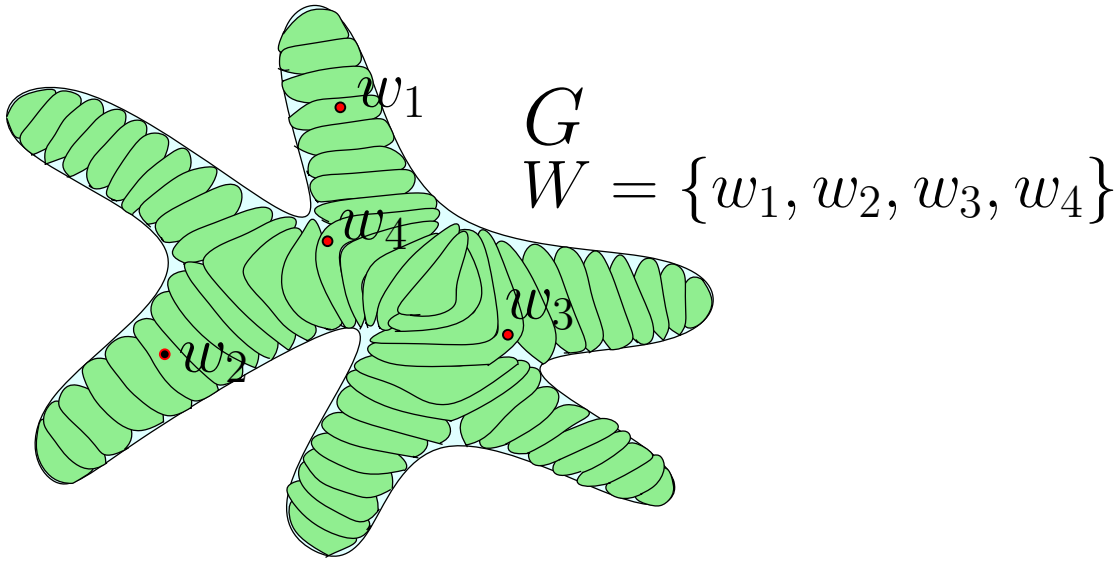
Let W be a largest bag of T SUBSET TREEWIDTH

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

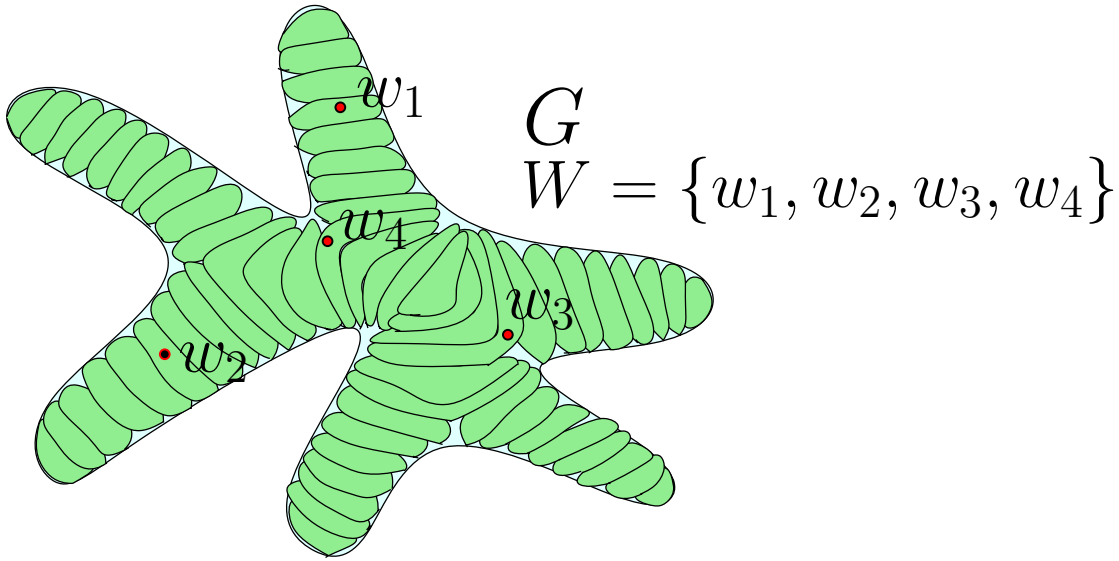
Let W be a largest bag of T SUBSET TREEWIDTH

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 $X \subset V(G)$



Improving a tree decomposition

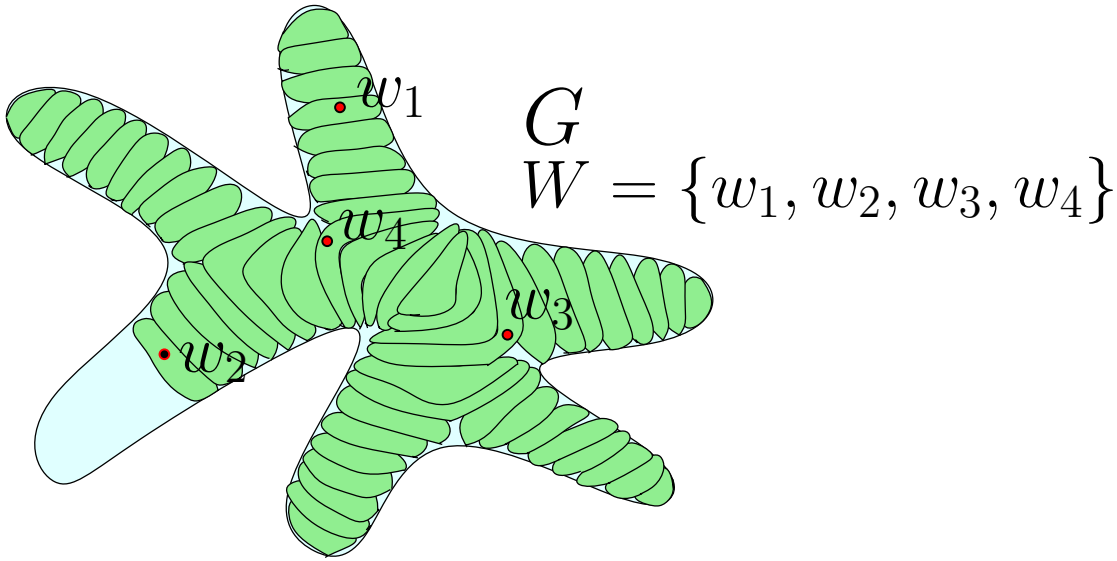
Let W be a largest bag of T SUBSET TREEWIDTH

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

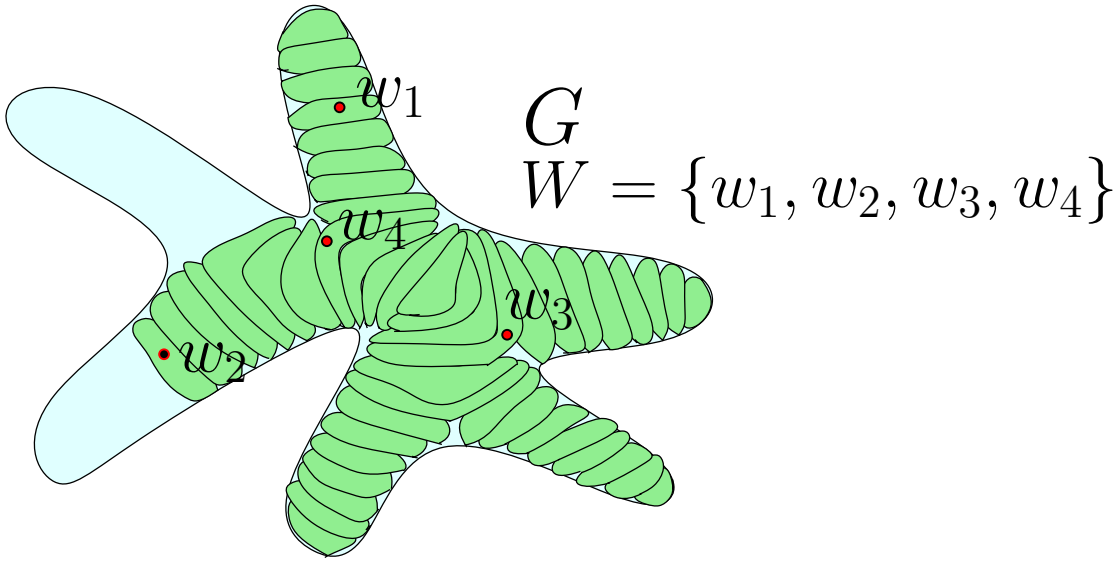
Let W be a largest bag of T SUBSET TREEWIDTH

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

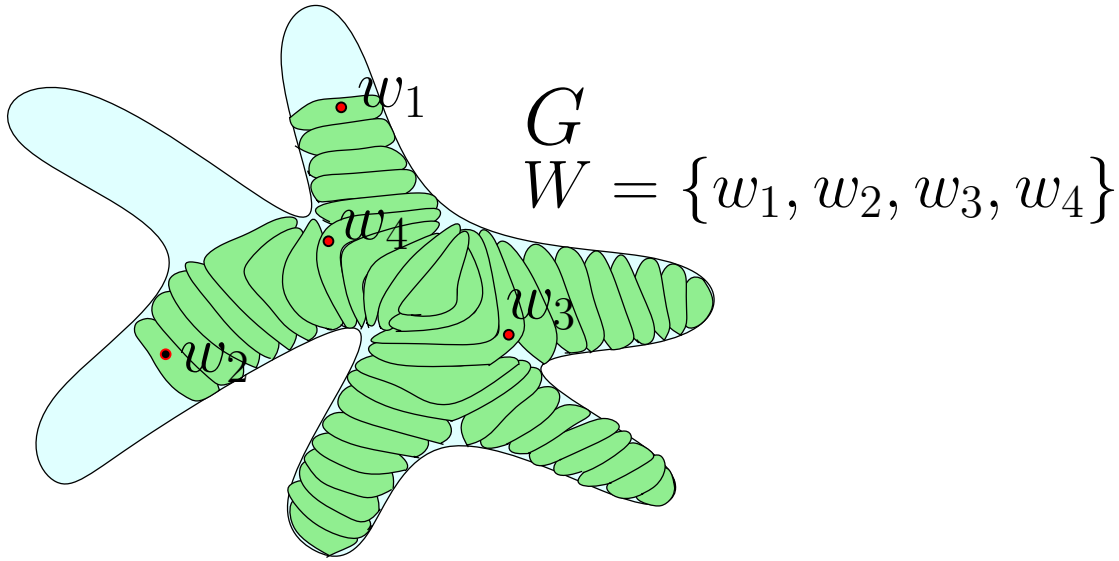
Let W be a largest bag of T SUBSET TREEWIDTH

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

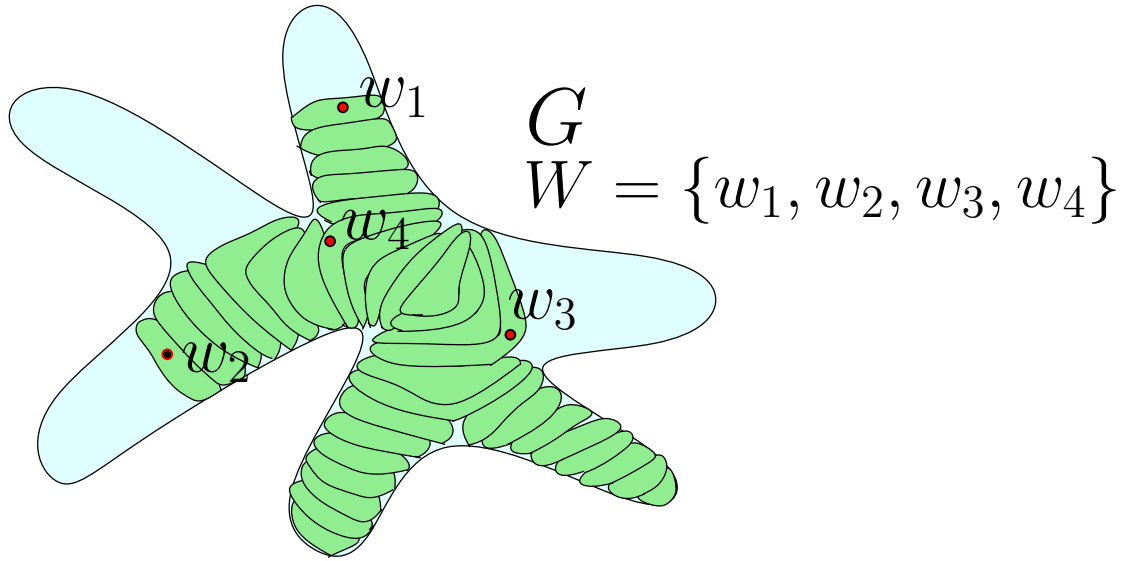
Let W be a largest bag of T SUBSET TREEWIDTH

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

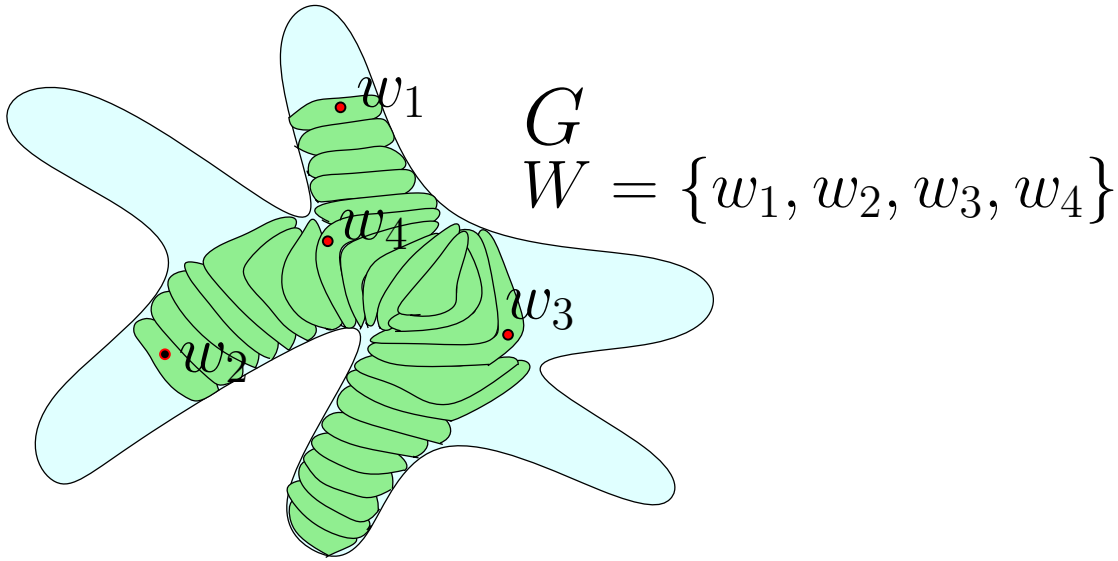
Let W be a largest bag of T — SUBSET TREEWIDTH

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

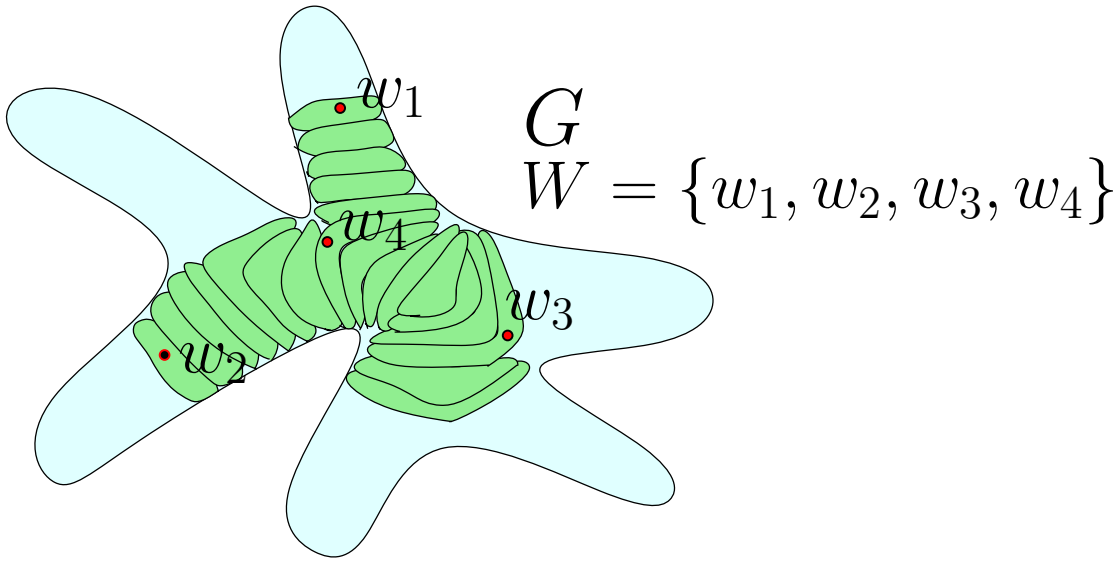
Let W be a largest bag of T SUBSET TREEWIDTH

Want to find:

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

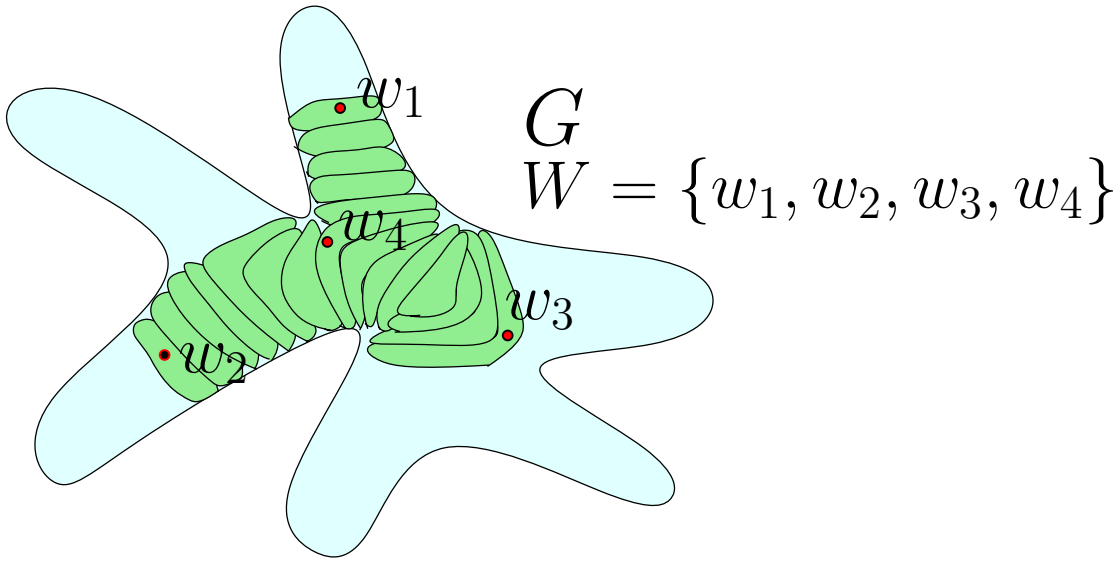
Let W be a largest bag of T SUBSET TREEWIDTH

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 $X \subset V(G)$



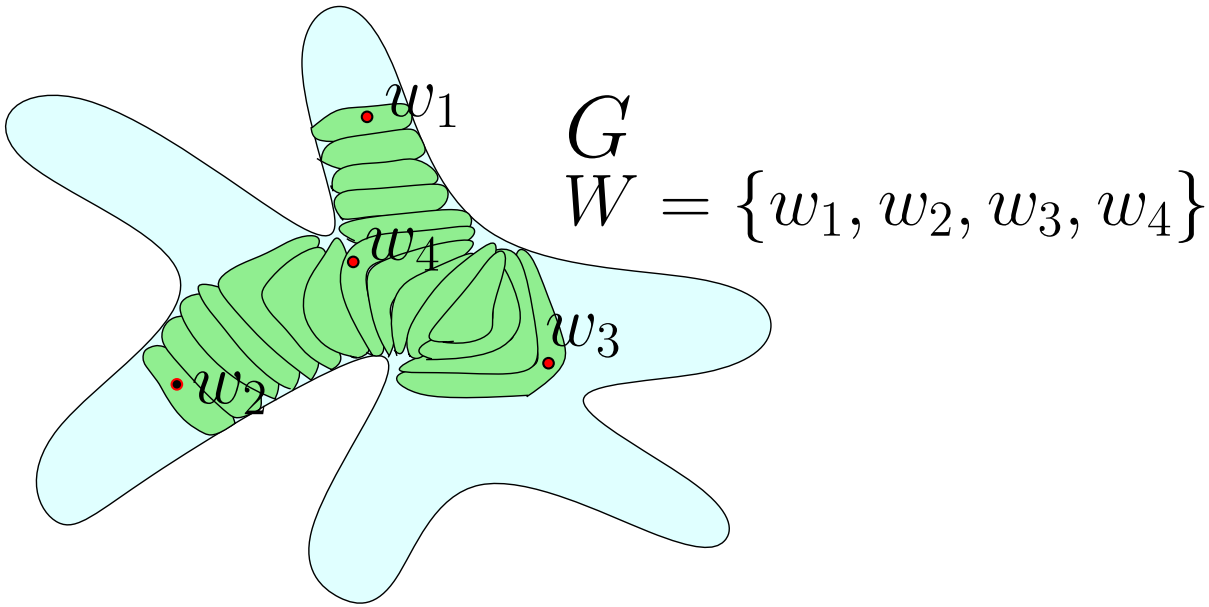
Improving a tree decomposition

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 $\leq |W| - 2$

Big-leaf formulation:



Improving a tree decomposition

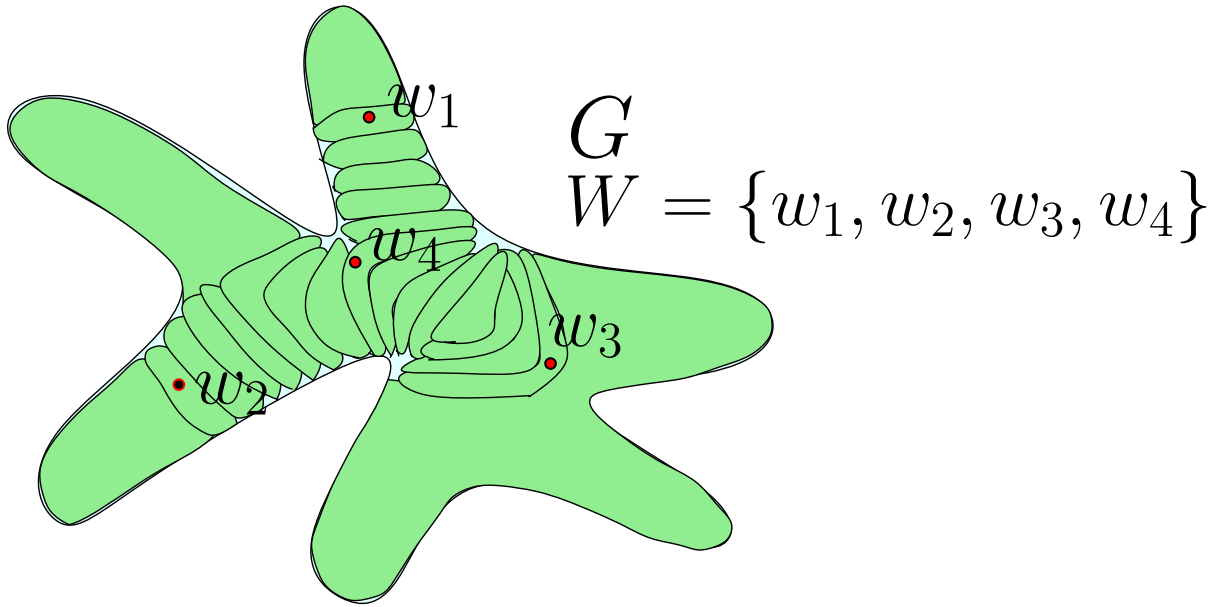
Let W be a largest bag of T SUBSET TREEWIDTH

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



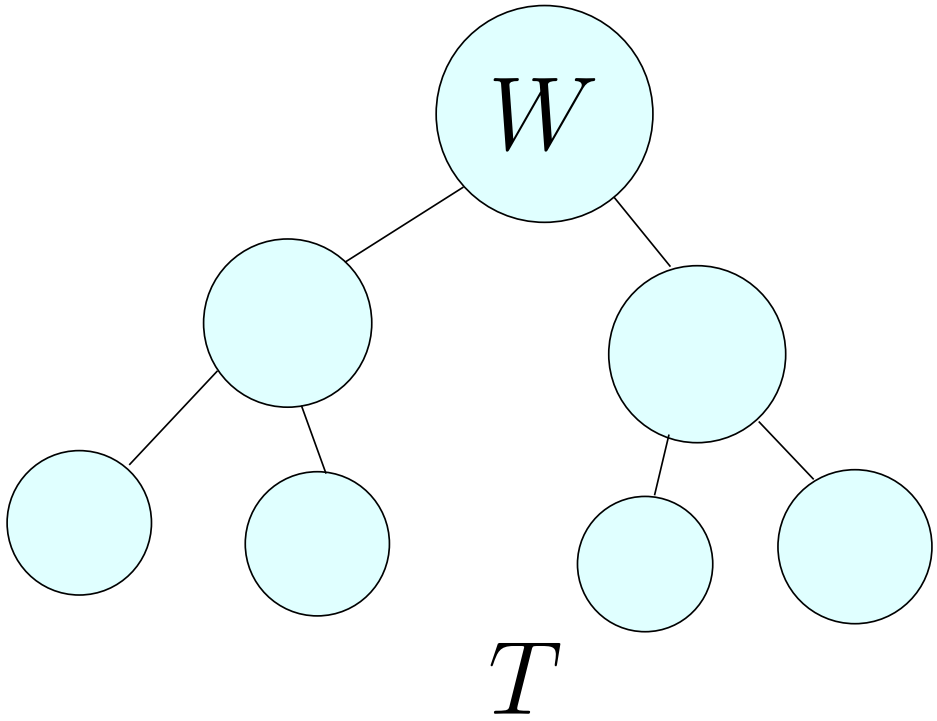
Improving a tree decomposition

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:



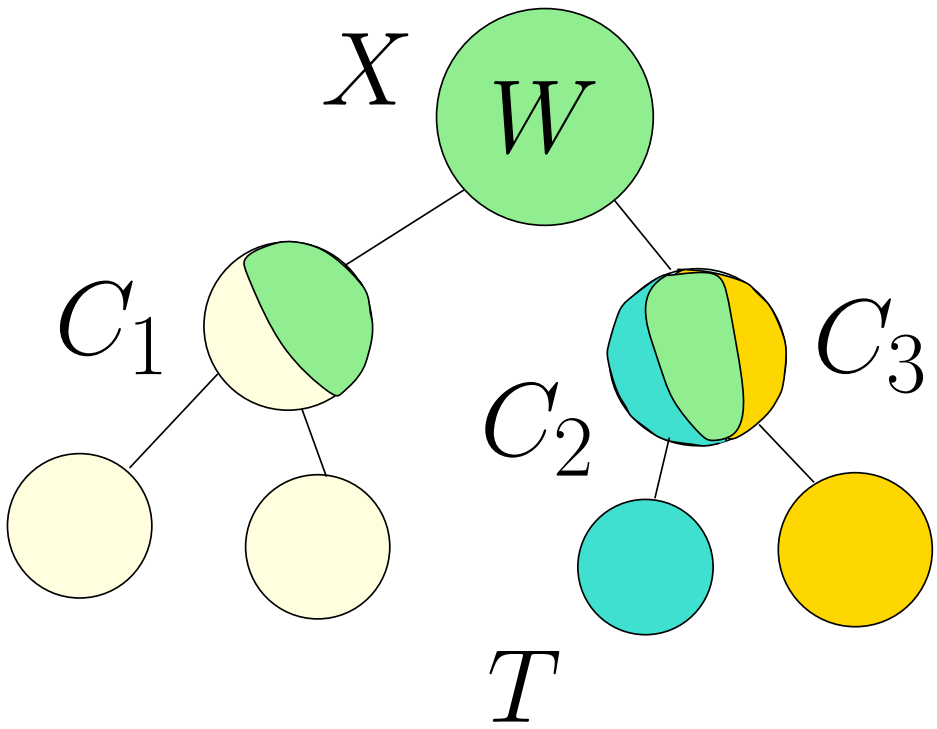
Improving a tree decomposition

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:



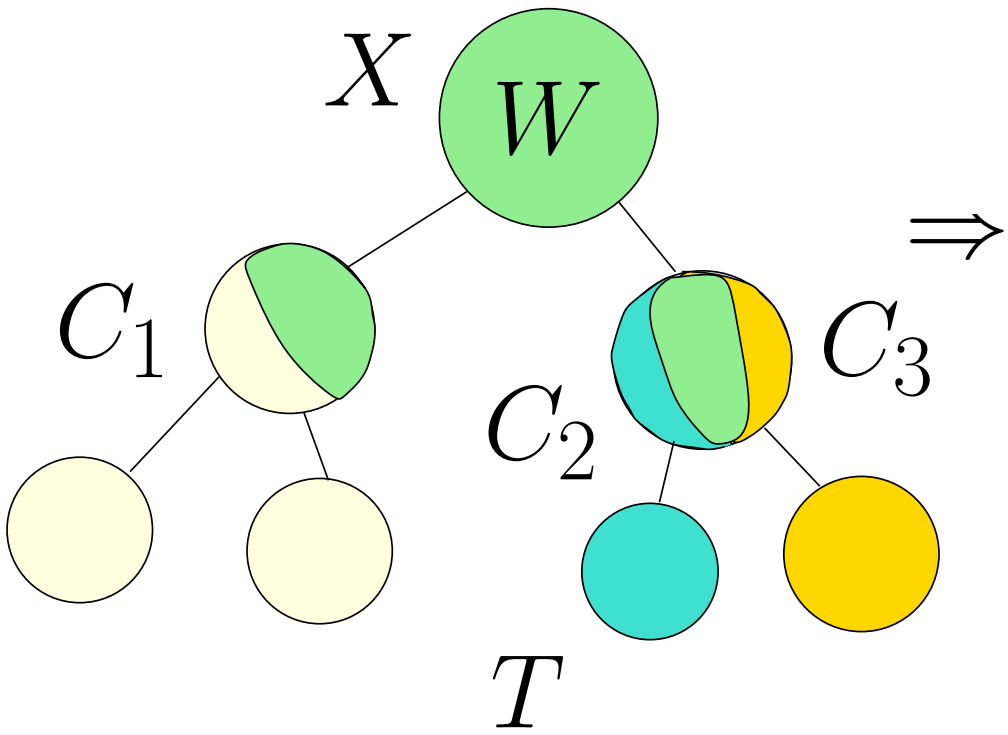
Improving a tree decomposition

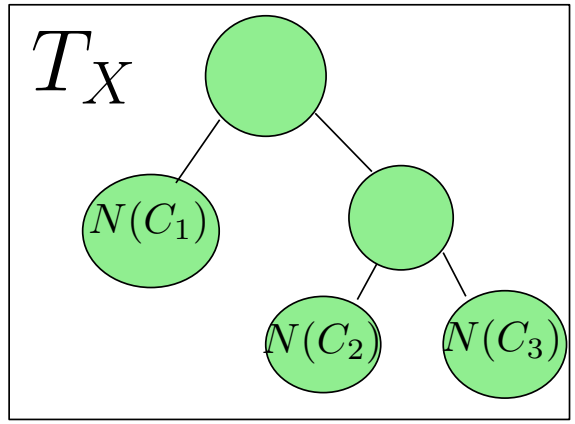
Let W be a largest bag of T SUBSET TREEWIDTH

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:





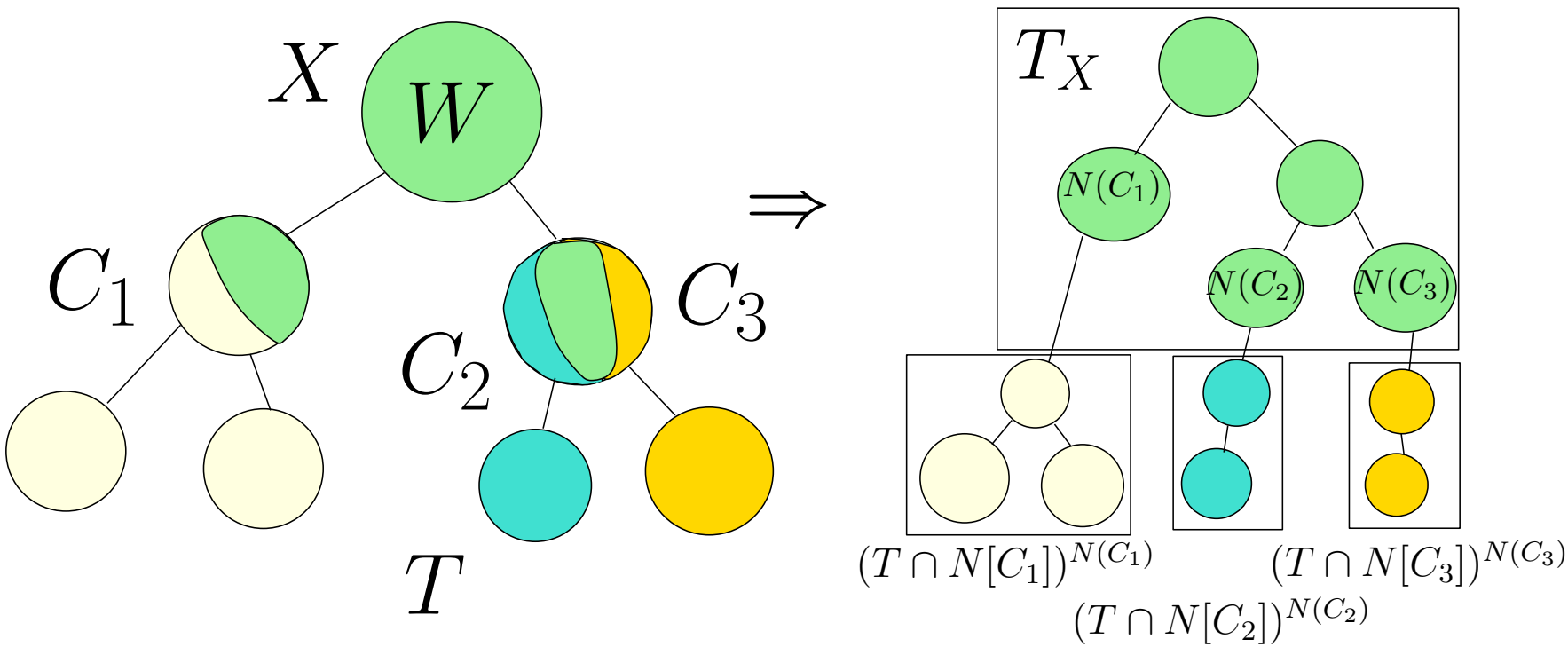
Improving a tree decomposition

Let W be a largest bag of T SUBSET TREEWIDTH

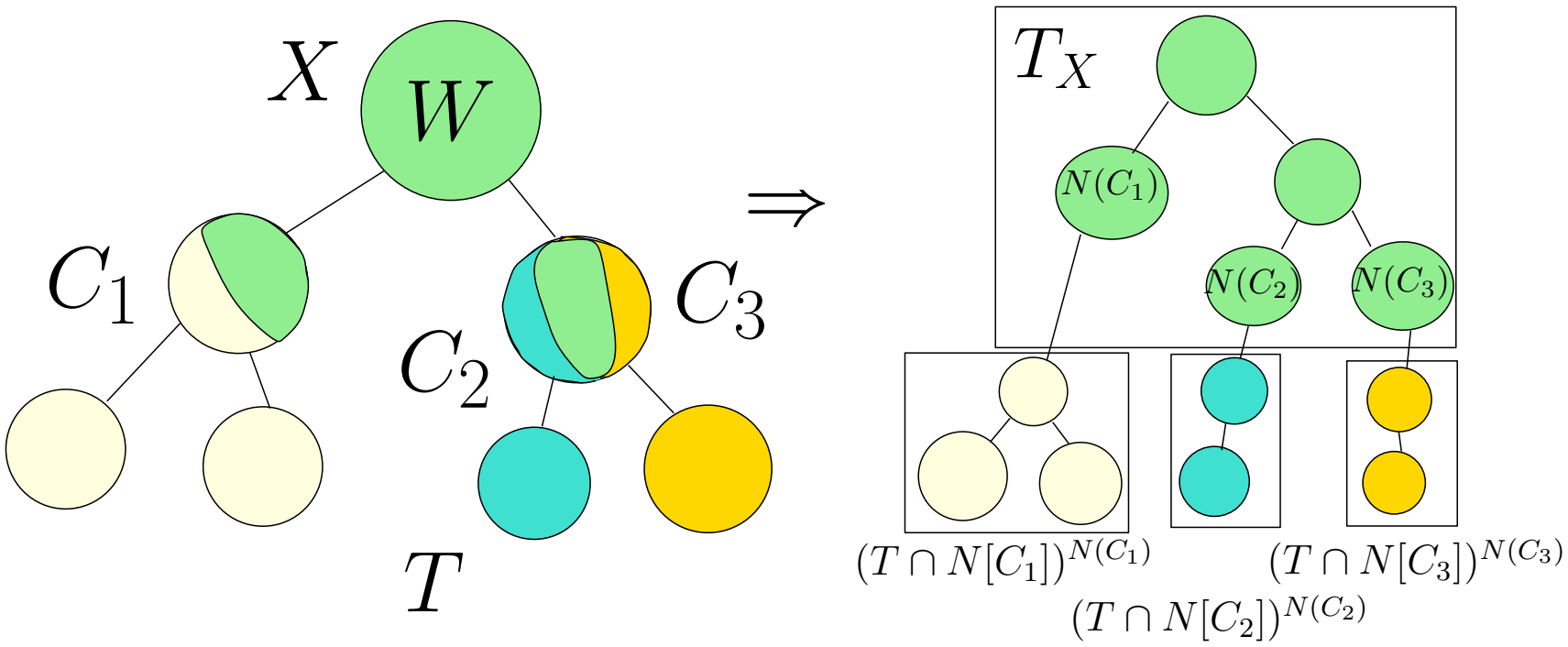
Have:

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

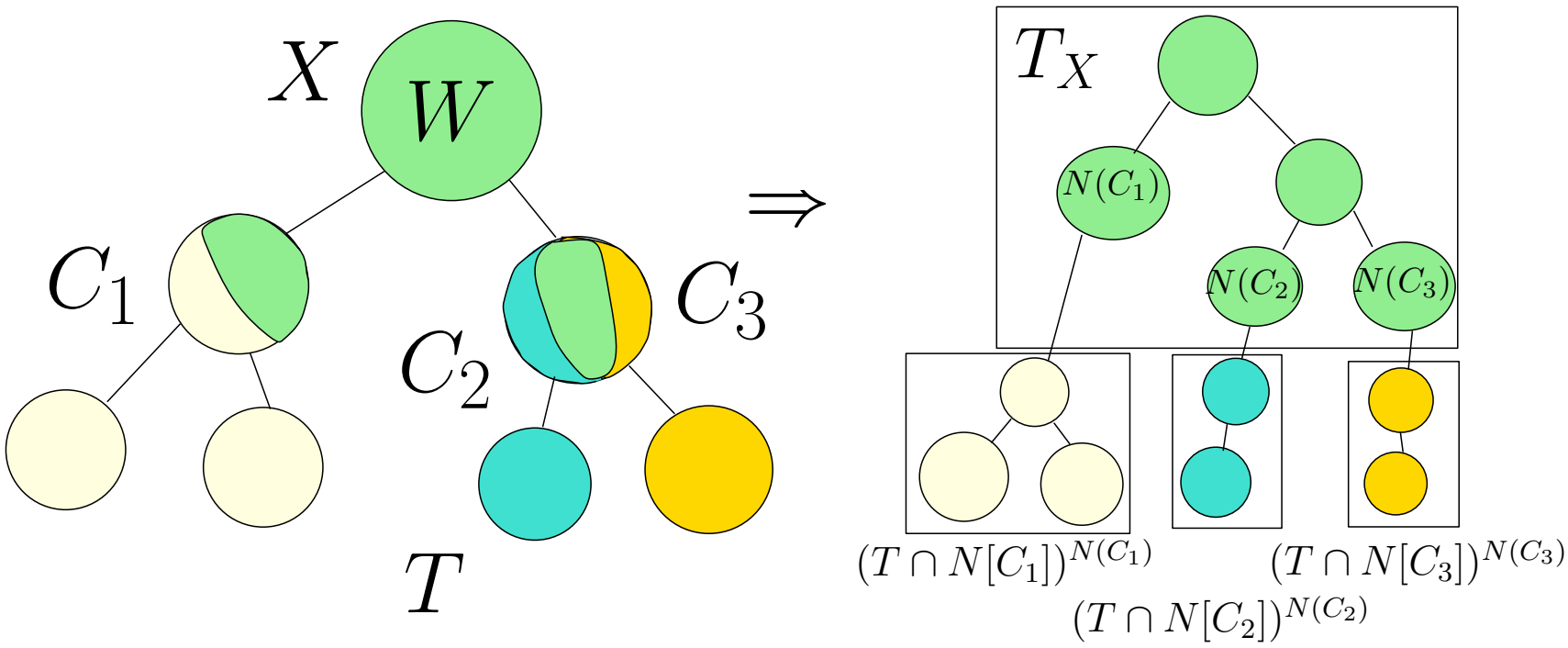
Improving T:



Does T improve?

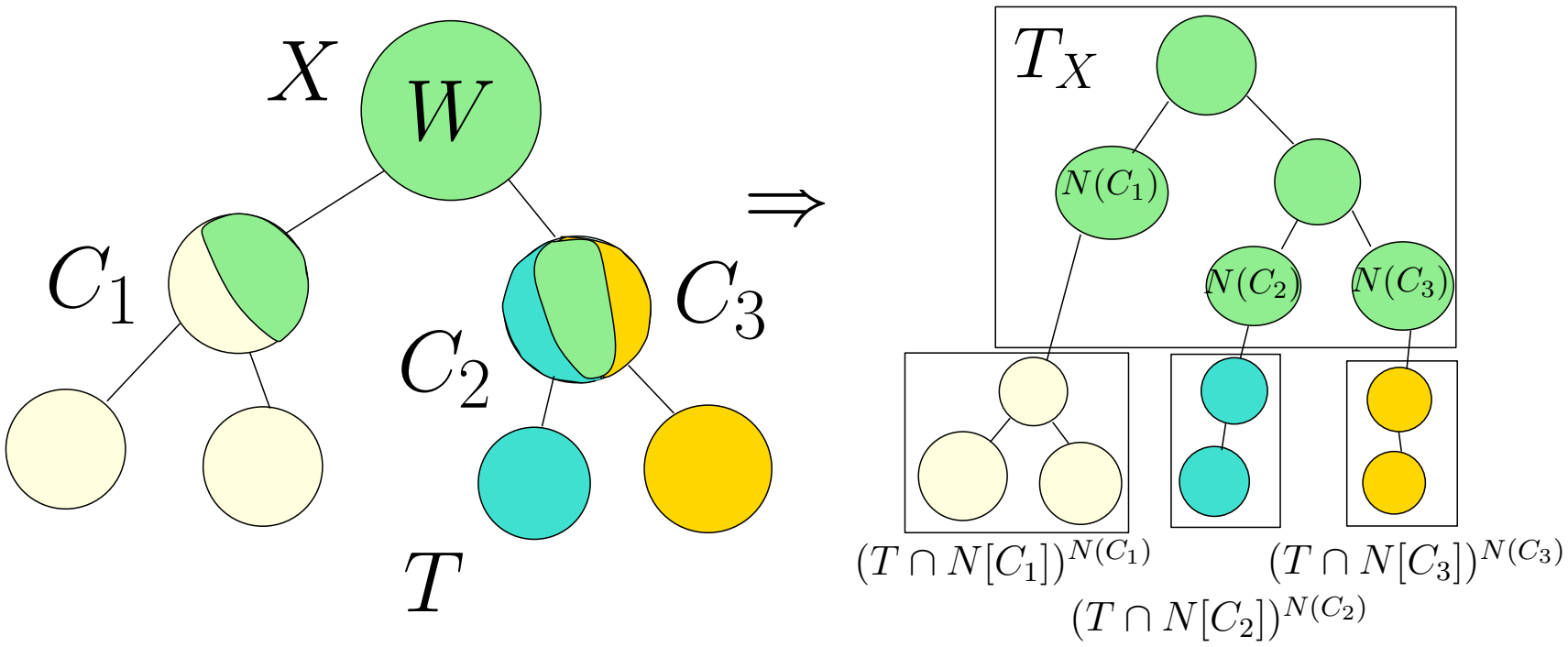


Does T improve?



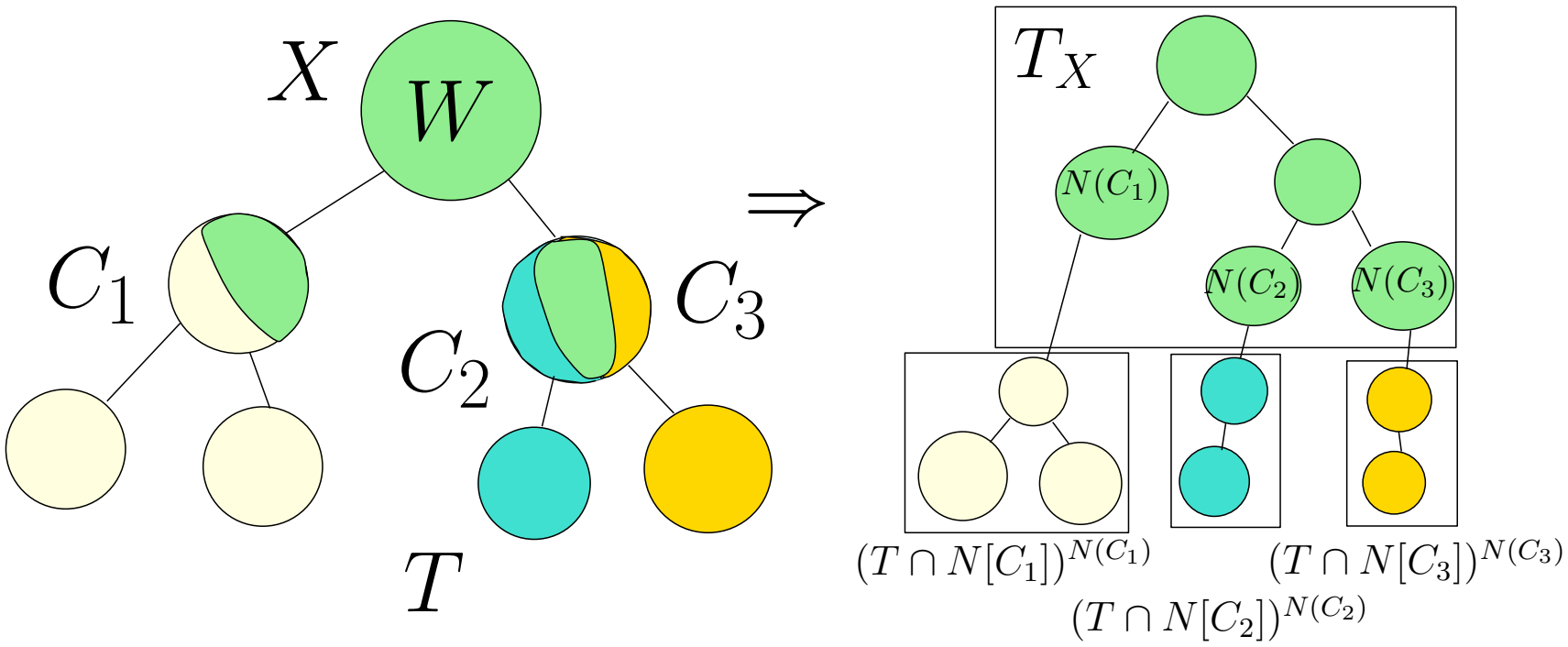
- Want: The copy of a bag in $(T \cap N[C_i])^{N(C_i)}$ is not larger than the original bag

Does T improve?



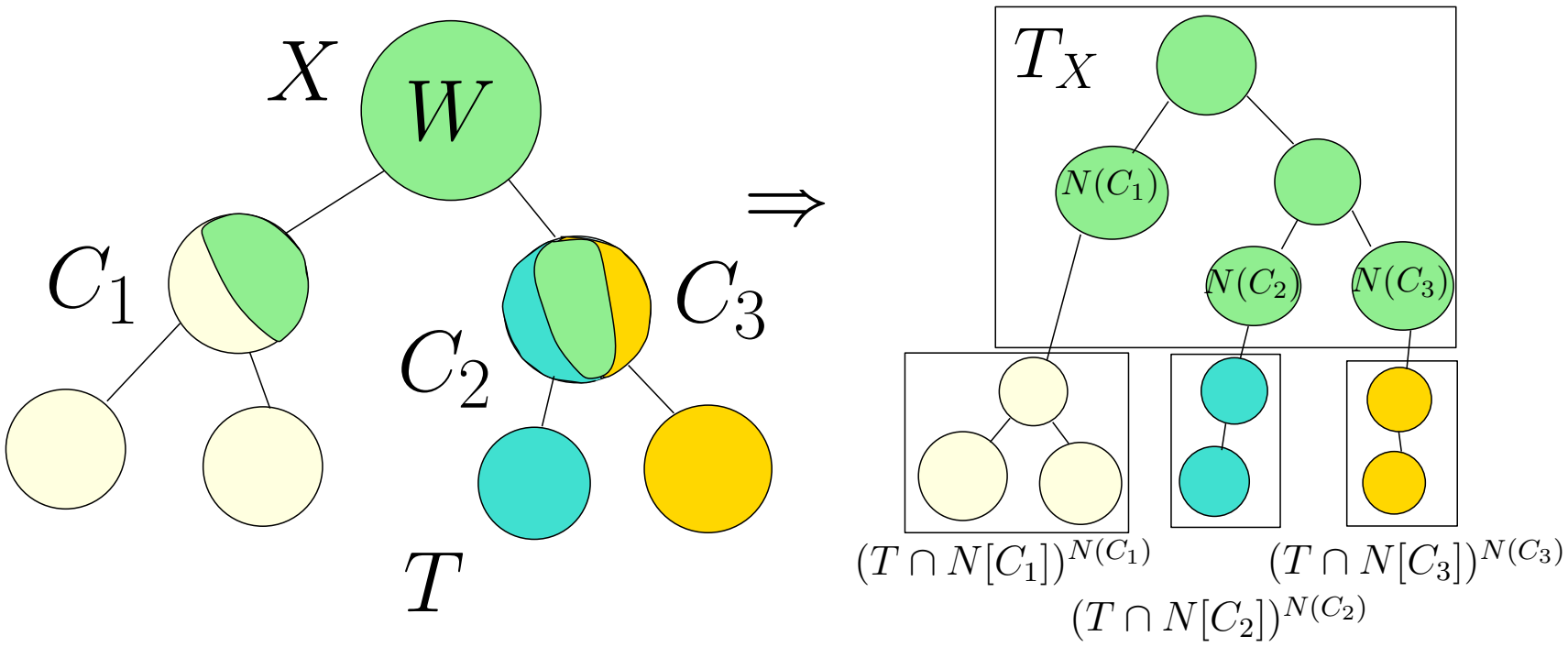
- Want: The copy of a bag in $(T \cap N[C_i])^{N(C_i)}$ is not larger than the original bag
- This holds if T_X is preprocessed so that its every bag is linked into W

Does T improve?



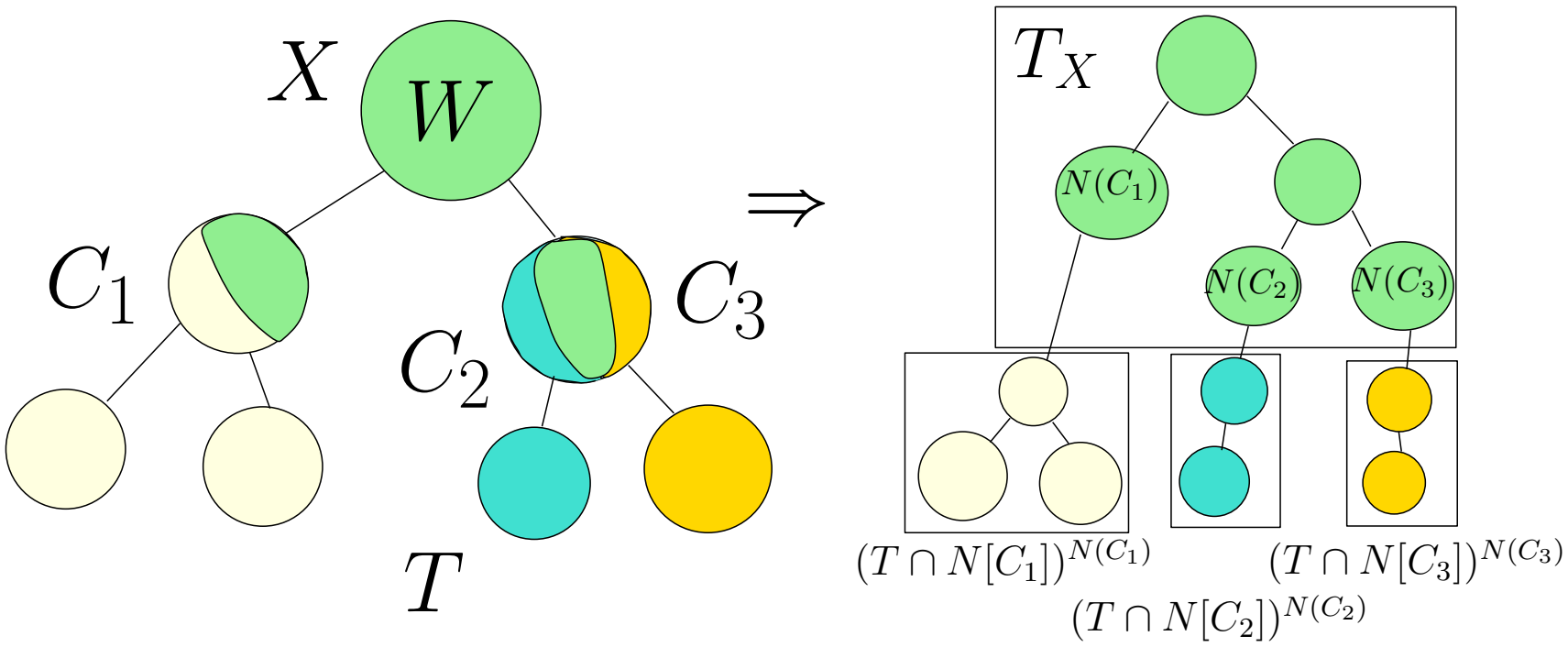
- Want: The copy of a bag in $(T \cap N[C_i])^{N(C_i)}$ is not larger than the original bag
- This holds if T_X is preprocessed so that its every bag is linked into W
  - $k^{\mathcal{O}(1)} n^4$ time here

Does T improve?



- Want: The copy of a bag in $(T \cap N[C_i])^{N(C_i)}$ is not larger than the original bag
- This holds if T_X is preprocessed so that its every bag is linked into W
  - $k^{\mathcal{O}(1)} n^4$ time here
- Proofs by Bellenbaum-Diestel type arguments

Does T improve?



- Want: The copy of a bag in $(T \cap N[C_i])^{N(C_i)}$ is not larger than the original bag
- This holds if T_X is preprocessed so that its every bag is linked into W
  - $k^{\mathcal{O}(1)} n^4$ time here
- Proofs by Bellenbaum-Diestel type arguments
- (actually needs a bit stronger condition than linkedness for improvement)

Subset treewidth for 2-approximation

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 $\leq |W| - 2$

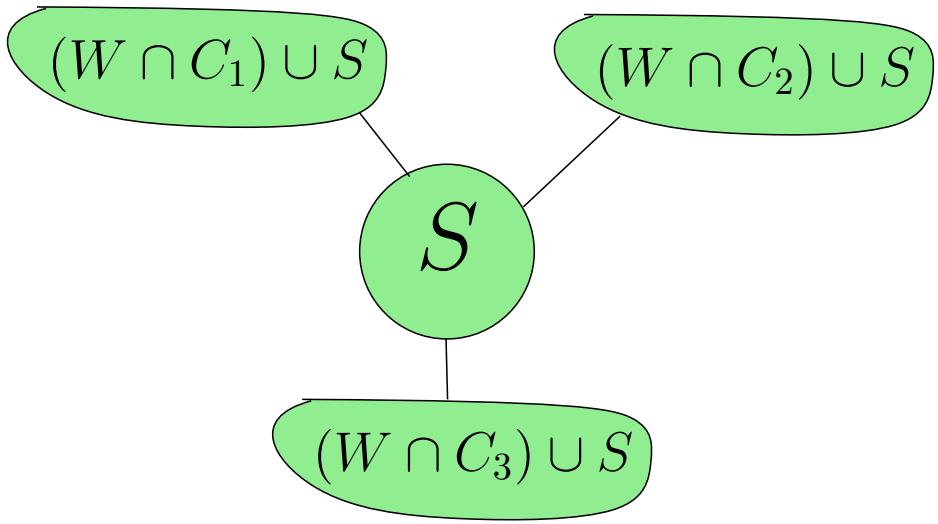
Subset treewidth for 2-approximation

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 $\leq |W| - 2$

If $|W| \geq 2 \cdot \text{tw}(G) + 3$, then such X and a decomposition of shape $K_{1,3}$ exists



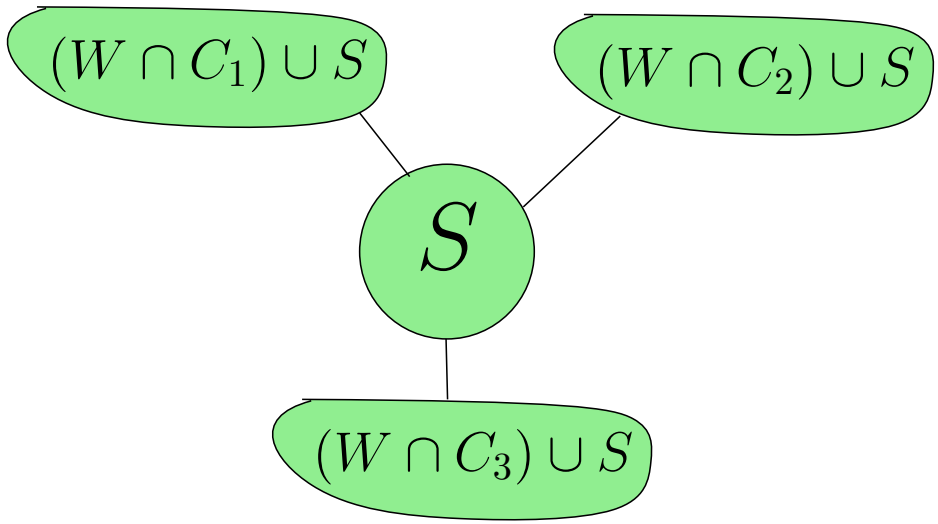
Subset treewidth for 2-approximation

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 $\leq |W| - 2$

If $|W| \geq 2 \cdot \text{tw}(G) + 3$, then such X and a decomposition of shape $K_{1,3}$ exists



- Where (S, C_1, C_2, C_3) is a balanced 3-way separation of G and $X = S \cup W$

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 $W_1, \ldots, W_t$

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 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

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

- Idea: Can afford to increase treewidth by εk

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

- Idea: Can afford to increase treewidth by εk
- Any set W can be partitioned into $t = \mathcal{O}(1/\varepsilon)$ sets $W_1, \ldots, W_t$ so that making them into cliques increases treewidth by at most $\varepsilon|W|$

Solving the subset treewidth problem

Solving the subset treewidth problem

Goal: Sketch $k^{\mathcal{O}(kt)}n^{\mathcal{O}(1)}$ time algorithm for partitioned subset treewidth

Solving the subset treewidth problem

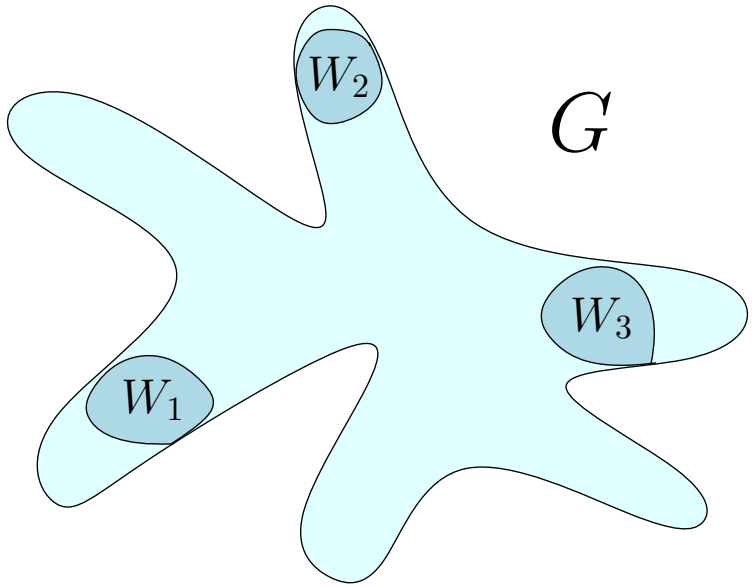
Goal: Sketch $k^{\mathcal{O}(kt)}n^{\mathcal{O}(1)}$ time algorithm for partitioned subset treewidth

(this is also a $k^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$ time algorithm for subset treewidth)

Solving subset treewidth

Setting:

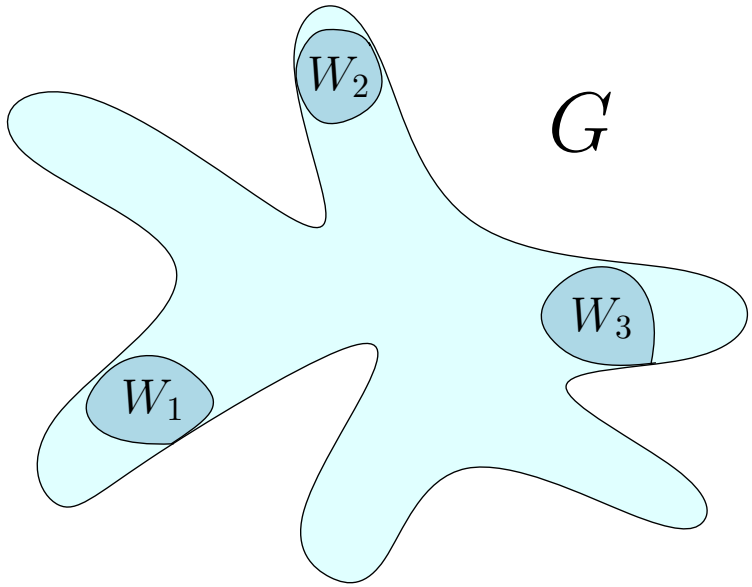
- Input: Graph G, t terminal cliques $W_1, \ldots, W_t$, and an integer k



Solving subset treewidth

Setting:

- Input: Graph G, t terminal cliques $W_1, \ldots, W_t$, and an integer k
- Goal: Find $X \supseteq \bigcup_{i=1}^{t} W_i$ and a tree decomposition of $\texttt{torso}(X)$ of width $\leq k$

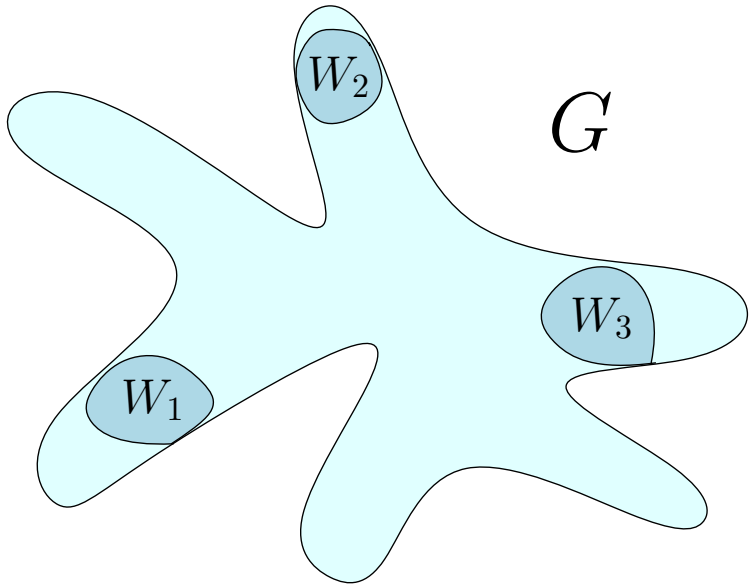


Solving subset treewidth

Setting:

- Input: Graph G, t terminal cliques $W_1, \ldots, W_t$, and an integer k
- Goal: Find $X \supseteq \bigcup_{i=1}^{t} W_i$ and a tree decomposition of $\texttt{torso}(X)$ of width $\leq k$

Reduction rule:



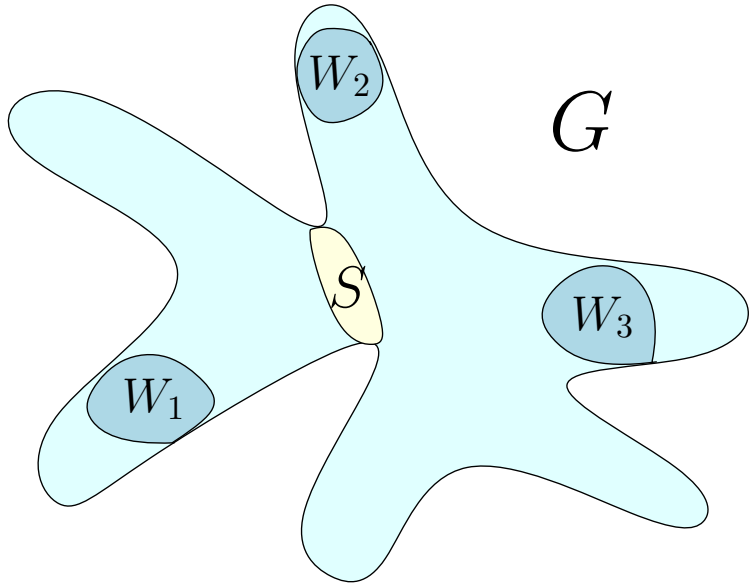
Solving subset treewidth

Setting:

- Input: Graph G, t terminal cliques $W_1, \ldots, W_t$, and an integer k
- Goal: Find $X \supseteq \bigcup_{i=1}^{t} W_i$ and a tree decomposition of $\texttt{torso}(X)$ of width $\leq k$

Reduction rule:

Let S be a non-trivial minimum size (W_i, W_j)-separator



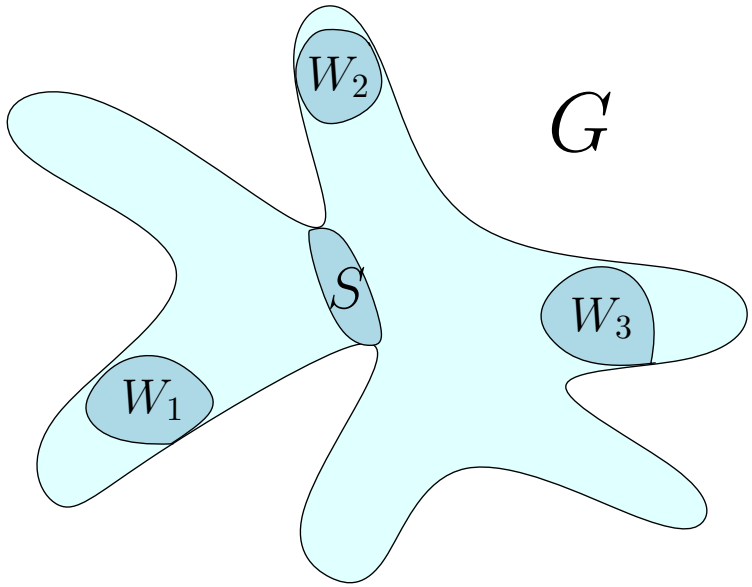
Solving subset treewidth

Setting:

- Input: Graph G, t terminal cliques $W_1, \ldots, W_t$, and an integer k
- Goal: Find $X \supseteq \bigcup_{i=1}^{t} W_i$ and a tree decomposition of $\texttt{torso}(X)$ of width $\leq k$

Reduction rule:

Let S be a non-trivial minimum size (W_i, W_j)-separator
Make S into a terminal clique and solve both sides independently



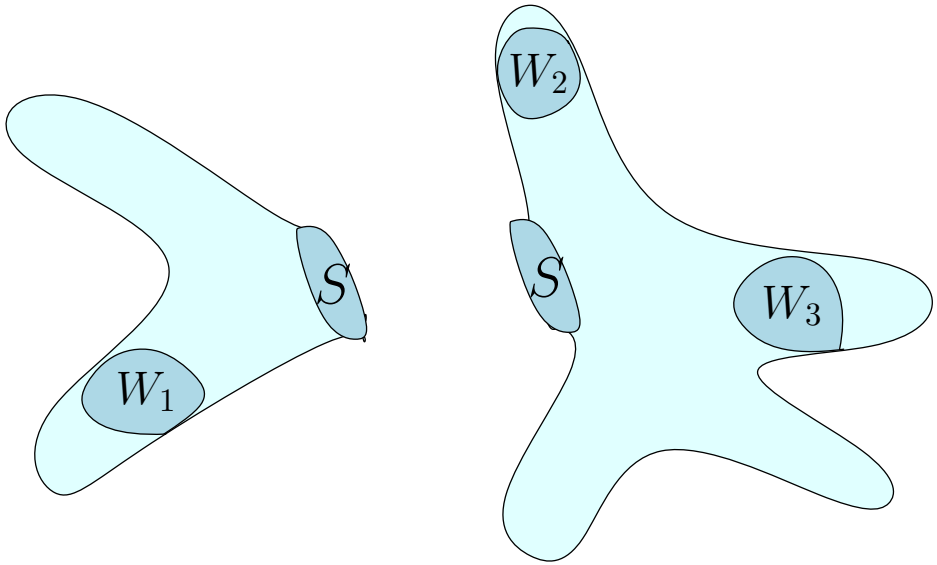
Solving subset treewidth

Setting:

- Input: Graph G, t terminal cliques $W_1, \ldots, W_t$, and an integer k
- Goal: Find $X \supseteq \bigcup_{i=1}^{t} W_i$ and a tree decomposition of $\texttt{torso}(X)$ of width $\leq k$

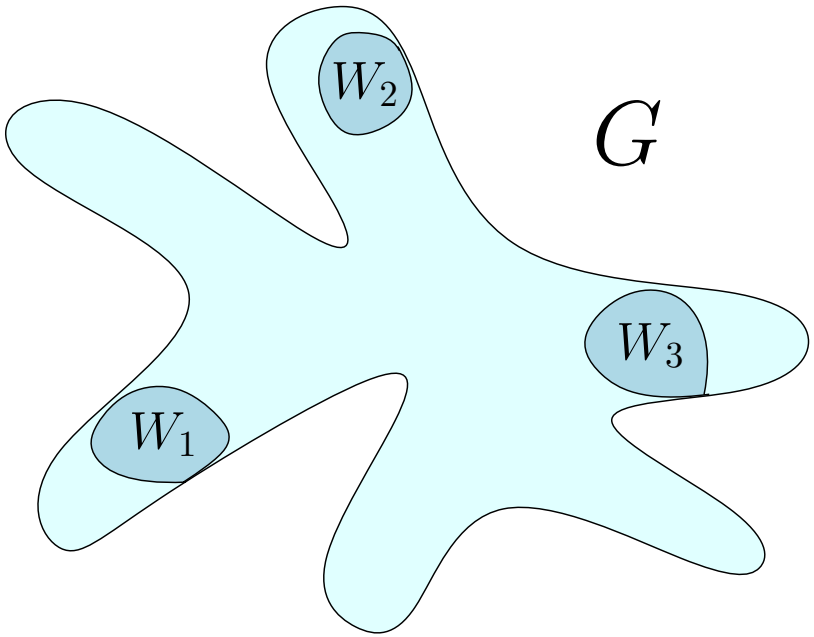
Reduction rule:

Let S be a non-trivial minimum size (W_i, W_j)-separator
Make S into a terminal clique and solve both sides independently



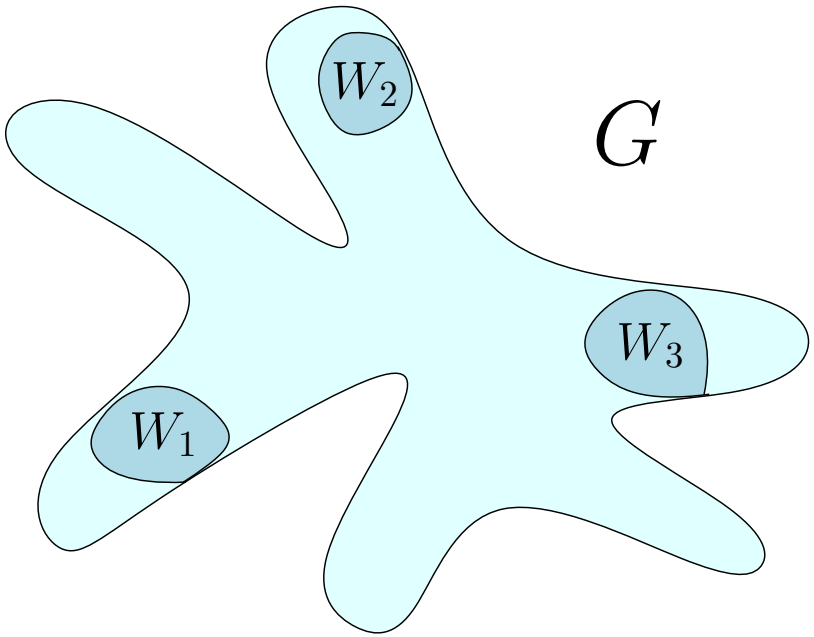
Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other



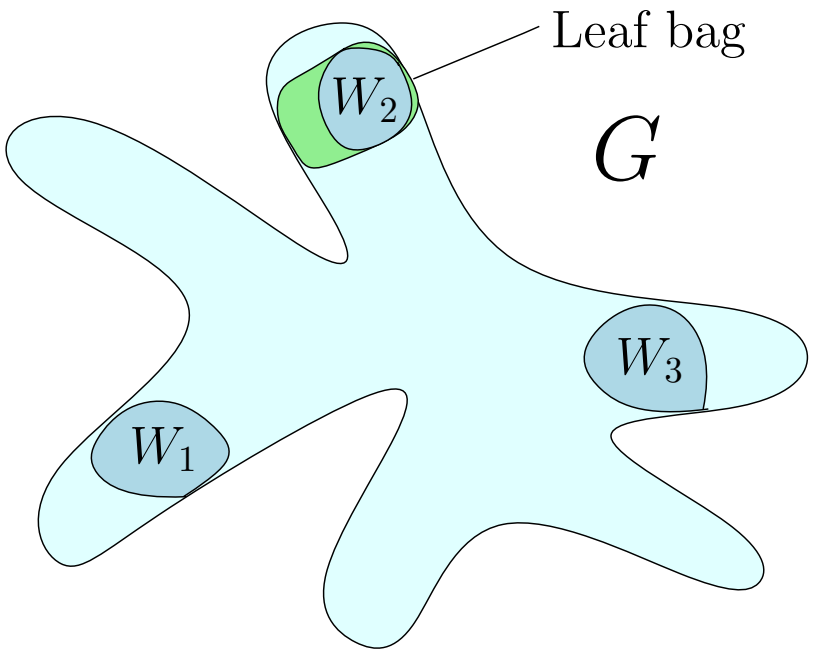
Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other
- Goal: To make progress, increase the size/flow of some terminal clique



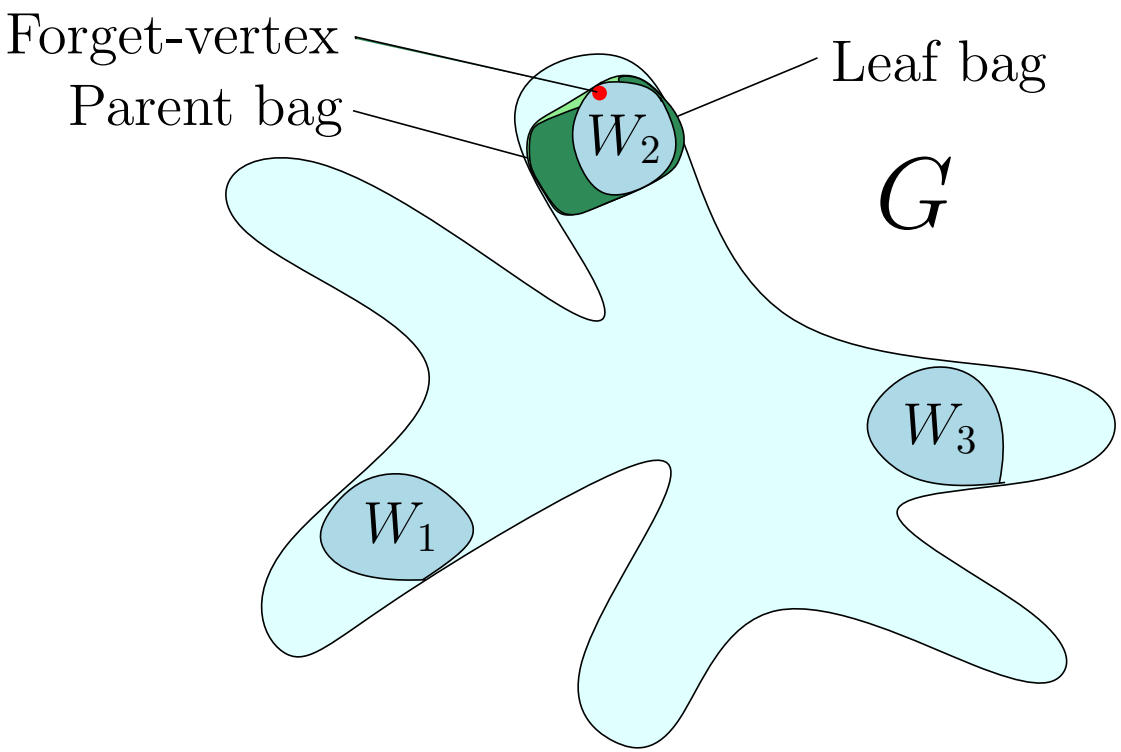
Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other
- Goal: To make progress, increase the size/flow of some terminal clique



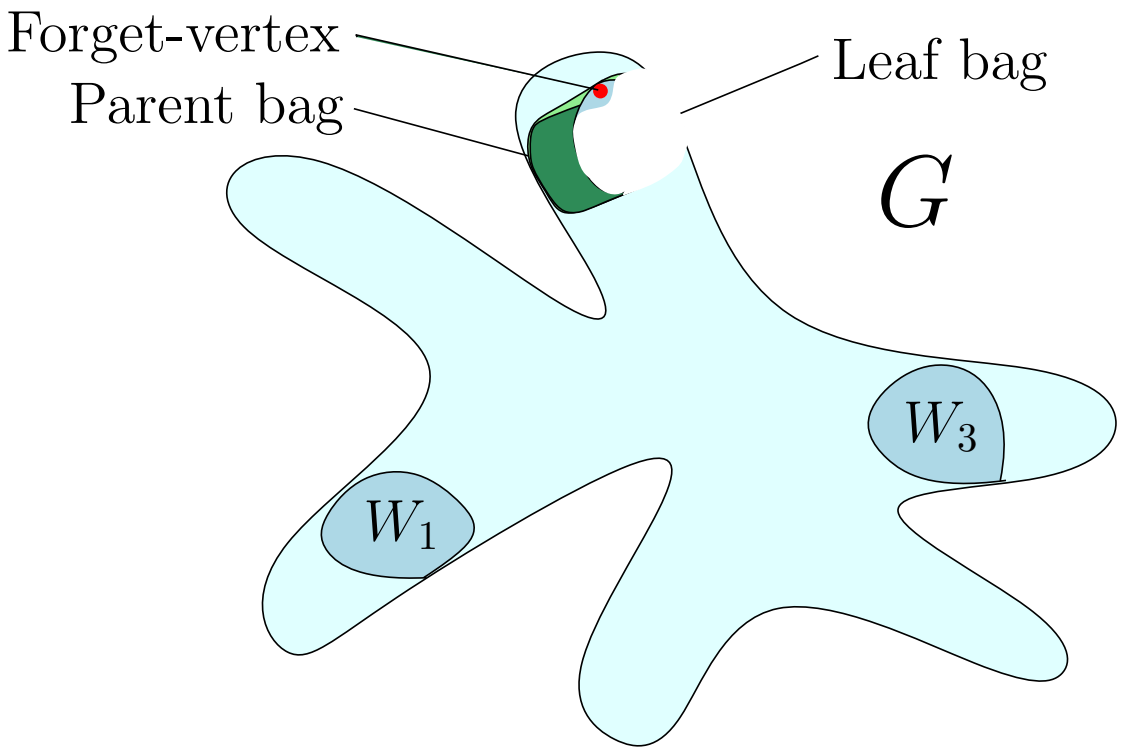
Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other
- Goal: To make progress, increase the size/flow of some terminal clique



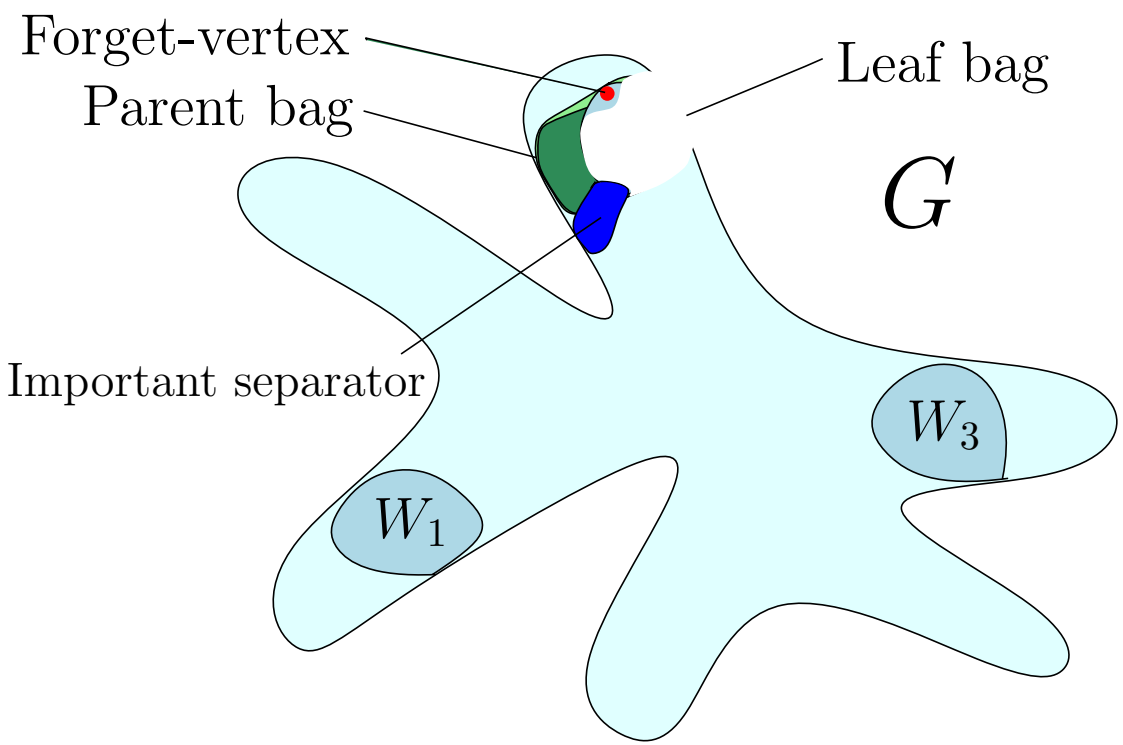
Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other
- Goal: To make progress, increase the size/flow of some terminal clique



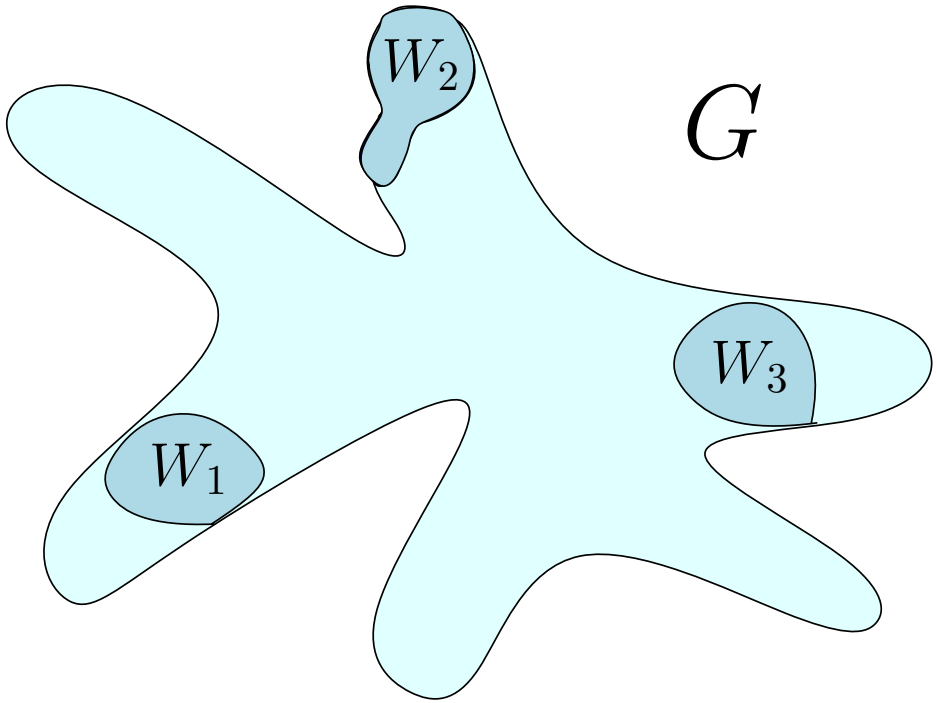
Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other
- Goal: To make progress, increase the size/flow of some terminal clique
- Increase W_2 by guessing an important separator

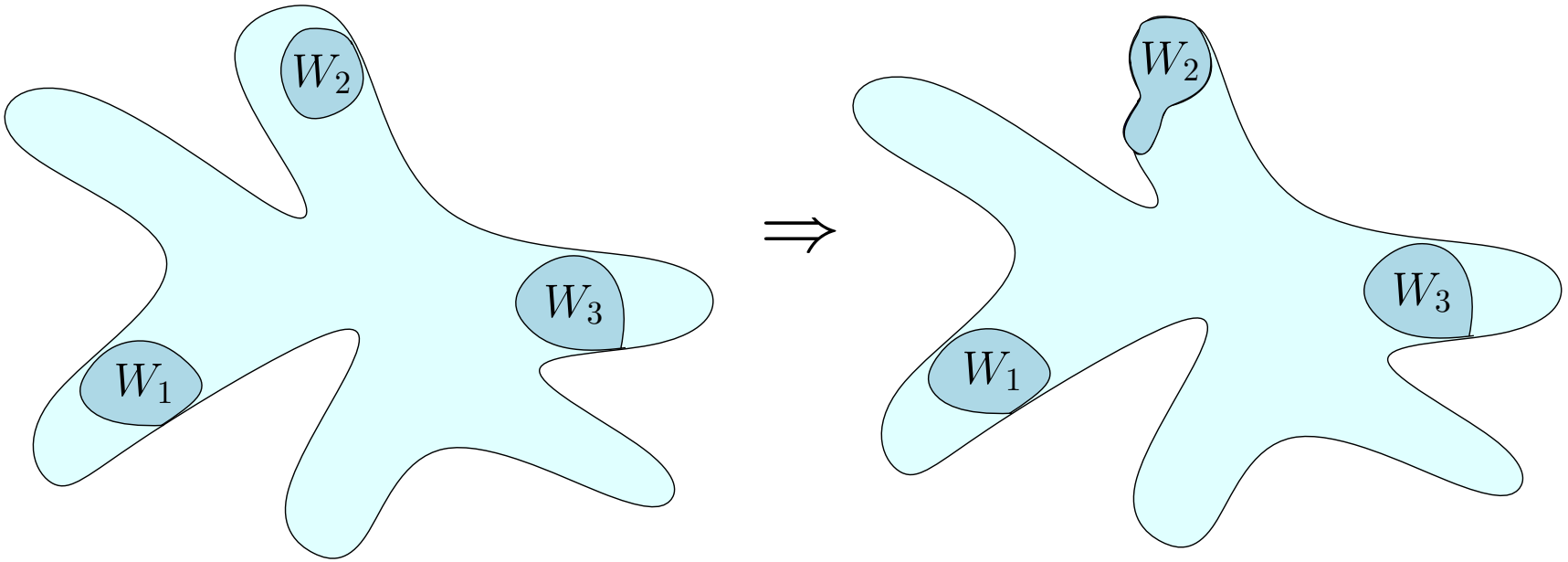


Branching for partitioned subset treewidth

- Now terminal cliques *strongly linked* into each other
- Goal: To make progress, increase the size/flow of some terminal clique
- Increase W_2 by guessing an important separator

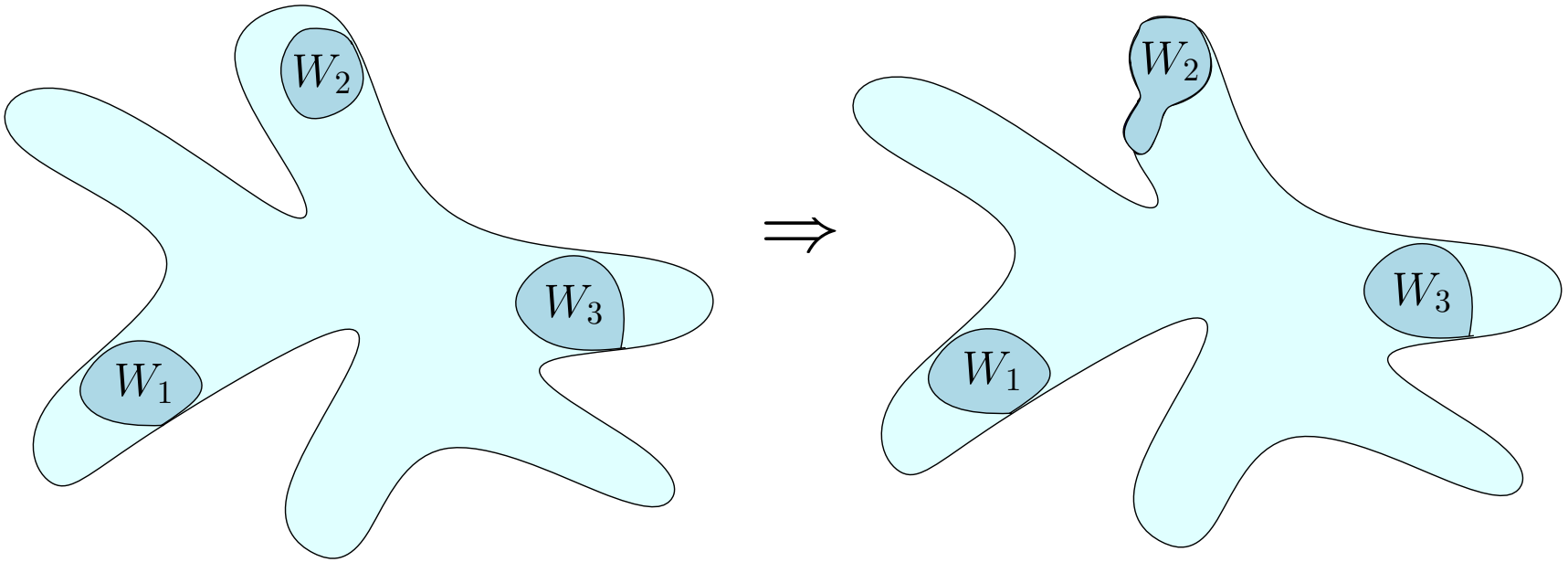


Analysis of branching



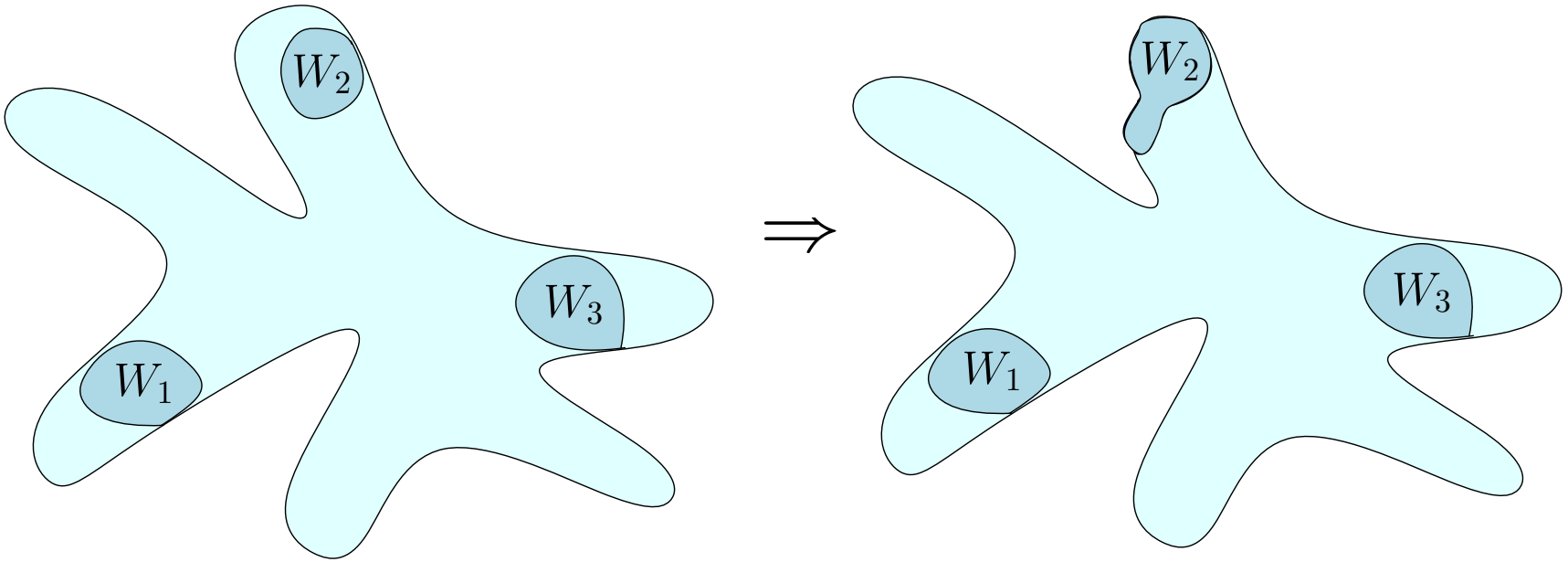
- Increased the size/flow of a leaf terminal clique by guessing a forget-vertex and an important separator

Analysis of branching



- Increased the size/flow of a leaf terminal clique by guessing a forget-vertex and an important separator
- Sum of sizes/flows of terminal cliques at most $(k+1)t$, so branching depth at most kt

Analysis of branching



- Increased the size/flow of a leaf terminal clique by guessing a forget-vertex and an important separator
- Sum of sizes/flows of terminal cliques at most $(k+1)t$, so branching depth at most kt
- To get $k^{\mathcal{O}(kt)}n^{\mathcal{O}(1)}$ time, need also an important separator hitting set lemma

Conclusion

Open questions:

Conclusion

Open questions:

- Is there $2^{\mathcal{O}(k^{1.999})}n^{\mathcal{O}(1)}$ time algorithm for subset treewidth?

Conclusion

Open questions:

- Is there $2^{\mathcal{O}(k^{1.999})}n^{\mathcal{O}(1)}$ time algorithm for subset treewidth?
- When $t = \mathcal{O}(1)$, is there $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm for partitioned subset treewidth?

Conclusion

Open questions:

- Is there $2^{\mathcal{O}(k^{1.999})}n^{\mathcal{O}(1)}$ time algorithm for subset treewidth?
- When $t = \mathcal{O}(1)$, is there $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm for partitioned subset treewidth?
- How much can the n^4 factor be optimized?

Thank you!

Thank you!