Linear-Time Algorithms for k-Edge-Connected Components, k-Lean Tree Decompositions, and More

Tuukka Korhonen

UNIVERSITY OF COPENHAGEN

10 October 2024

k-edge-connected components

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)} m$ time algorithm for k-edge-connected components

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)} m$ time algorithm for k-edge-connected components

Previous work:

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)} m$ time algorithm for k-edge-connected components

Previous work:

- $\mathcal{O}(m)$ for $k = 2$ [Hopcroft & Tarjan '73]

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)} m$ time algorithm for k-edge-connected components

Previous work:

- $\mathcal{O}(m)$ for $k = 2$ [Hopcroft & Tarjan '73]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91]

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)}m$ time algorithm for k-edge-connected components

Previous work:

- $\mathcal{O}(m)$ for $k = 2$ [Hopcroft & Tarjan '73]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91]
- $\mathcal{O}(m)$ for $k = 4$ [Nadara, Radecki, Smulewicz, Sokolowski'21, Georgiadis, Italiano, Kosinas'21]

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)}m$ time algorithm for k-edge-connected components

Previous work:

- $\mathcal{O}(m)$ for $k = 2$ [Hopcroft & Tarjan '73]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91]
- $\mathcal{O}(m)$ for $k = 4$ [Nadara, Radecki, Smulewicz, Sokolowski'21, Georgiadis, Italiano, Kosinas'21]
- $\mathcal{O}(m)$ for $k = 5$ [Kosinas '24]

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)} m$ time algorithm for k-edge-connected components

Previous work:

- $\mathcal{O}(m)$ for $k = 2$ [Hopcroft & Tarjan '73]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91]
- $\mathcal{O}(m)$ for $k = 4$ [Nadara, Radecki, Smulewicz, Sokolowski'21, Georgiadis, Italiano, Kosinas'21]
- $\mathcal{O}(m)$ for $k = 5$ [Kosinas '24]
- $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ for all k [Hariharan, Kavitha, Panigrahi '07]

k-edge-connected components

Def: Vertices u and v in the same k-edge-connected component if no u-v cut with $< k$ edges

Obs: This gives equivalence relation among vertices $\Rightarrow$ unique partition into components

Theorem (This work)

There is $k^{\mathcal{O}(k^2)} m$ time algorithm for k-edge-connected components

Previous work:

- $\mathcal{O}(m)$ for $k = 2$ [Hopcroft & Tarjan '73]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91]
- $\mathcal{O}(m)$ for $k = 4$ [Nadara, Radecki, Smulewicz, Sokolowski'21, Georgiadis, Italiano, Kosinas'21]
- $\mathcal{O}(m)$ for $k = 5$ [Kosinas '24]
- $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ for all k [Hariharan, Kavitha, Panigrahi '07]

For minimum cut:

- $\mathcal{O}(k^2 m \log m)$ [Gabow '91], $\mathcal{O}(m \operatorname{polylog} m)$ [Karger '96]

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$
- **Menger-like property:**

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$
- **Menger-like property:**

Let $t_1, t_2 \in V(T)$ and $X_1 \subseteq \text{bag}(t_1)$, $X_2 \subseteq \text{bag}(t_2)$ with $|X_1| = |X_2| \leq k$, then:

- Unless there is t_1-t_2-adhesion of size $< |X_1|$, there are $|X_1|$ vertex-disjoint paths linking X_1 to X_2

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$
- **Menger-like property:**

Let $t_1, t_2 \in V(T)$ and $X_1 \subseteq \text{bag}(t_1)$, $X_2 \subseteq \text{bag}(t_2)$ with $|X_1| = |X_2| \leq k$, then:

- Unless there is t_1-t_2-adhesion of size $< |X_1|$, there are $|X_1|$ vertex-disjoint paths linking X_1 to X_2

Theorem (This work)

There is a $k^{\mathcal{O}(k^2)} m$ time algorithm for computing a k-lean tree decomposition

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$
- **Menger-like property:**

Let $t_1, t_2 \in V(T)$ and $X_1 \subseteq \text{bag}(t_1)$, $X_2 \subseteq \text{bag}(t_2)$ with $|X_1| = |X_2| \leq k$, then:

- Unless there is t_1-t_2-adhesion of size $< |X_1|$, there are $|X_1|$ vertex-disjoint paths linking X_1 to X_2

Theorem (This work)

There is a $k^{\mathcal{O}(k^2)} m$ time algorithm for computing a k-lean tree decomposition

Obs: k-lean tree decomposition is (i, i)-unbreakable for all $i \leq k$

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$
- **Menger-like property:**

Let $t_1, t_2 \in V(T)$ and $X_1 \subseteq \text{bag}(t_1)$, $X_2 \subseteq \text{bag}(t_2)$ with $|X_1| = |X_2| \leq k$, then:

- Unless there is t_1-t_2-adhesion of size $< |X_1|$, there are $|X_1|$ vertex-disjoint paths linking X_1 to X_2

Theorem (This work)

There is a $k^{\mathcal{O}(k^2)} m$ time algorithm for computing a k-lean tree decomposition

Obs: k-lean tree decomposition is (i, i)-unbreakable for all $i \leq k$

$\Rightarrow$ Linear-time FPT algorithm for unbreakable decomposition with optimal unbreakability parameters

k-lean tree decompositions

Tree decomposition (T, bag) of a graph G with

- Adhesion size $< k$
- **Menger-like property:**

Let $t_1, t_2 \in V(T)$ and $X_1 \subseteq \text{bag}(t_1)$, $X_2 \subseteq \text{bag}(t_2)$ with $|X_1| = |X_2| \leq k$, then:

- Unless there is t_1-t_2-adhesion of size $< |X_1|$, there are $|X_1|$ vertex-disjoint paths linking X_1 to X_2

Theorem (This work)

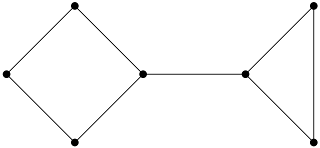
There is a $k^{\mathcal{O}(k^2)} m$ time algorithm for computing a k-lean tree decomposition

Obs: k-lean tree decomposition is (i, i)-unbreakable for all $i \leq k$

$\Rightarrow$ Linear-time FPT algorithm for unbreakable decomposition with optimal unbreakability parameters

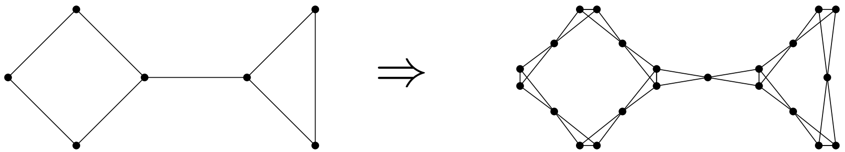
- Improves upon [Anand, Lee, Li, Long, Saranurak '25], but with worse $f(k)$ in the running time

Reducing k-edge-connected components to k-lean tree decomposition



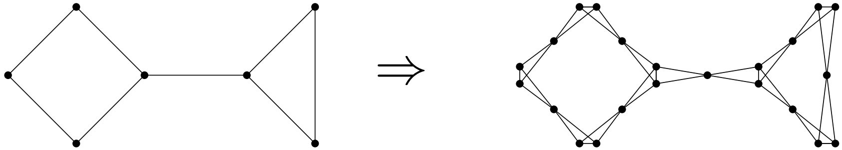
Reducing k-edge-connected components to k-lean tree decomposition

- Replace vertices by cliques of size k
- Create vertex for each edge and connect to the cliques corresponding to its endpoints



Reducing k-edge-connected components to k-lean tree decomposition

- Replace vertices by cliques of size k
- Create vertex for each edge and connect to the cliques corresponding to its endpoints
- Resulting k-lean tree decomposition gives k-Gomory-Hu tree



The algorithm

The algorithm

Part 1: Proof that “improver algorithm” implies the algorithm

The algorithm

Part 1: Proof that “improver algorithm” implies the algorithm (Inspired by [Bodlaender ’93])

The algorithm

Part 1: Proof that “improver algorithm” implies the algorithm (Inspired by [Bodlaender ’93])

Part 2: The improver algorithm

The algorithm

Part 1: Proof that "improver algorithm" implies the algorithm (Inspired by [Bodlaender '93])

Part 2: The improver algorithm (Inspired by [Graph Minors X., Robertson & Seymour '91])

Part 1: Improver algorithm implies the algorithm

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: Tree decomposition with

- Adhesion size $< 2k$
- $(2k, k)$-unbreakable bags

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: Tree decomposition with

- Adhesion size $< 2k$
- $(2k, k)$-unbreakable bags

Output: k-lean tree decomposition

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: Tree decomposition with

- Adhesion size $< 2k$
- $(2k, k)$-unbreakable bags

Output: k-lean tree decomposition

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$
  - Call the algorithm recursively on G/M

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$
  - Call the algorithm recursively on G/M
  - "Uncontract" the k-lean tree decomposition of G/M to get a tree decomposition of G with adhesion size $< 2k$ and $(2k, k)$-unbreakable bags

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$
  - Call the algorithm recursively on G/M
  - "Uncontract" the k-lean tree decomposition of G/M to get a tree decomposition of G with adhesion size $< 2k$ and $(2k, k)$-unbreakable bags
  - Apply the improver algorithm and return

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$
  - Call the algorithm recursively on G/M
  - "Uncontract" the k-lean tree decomposition of G/M to get a tree decomposition of G with adhesion size $< 2k$ and $(2k, k)$-unbreakable bags
  - Apply the improver algorithm and return
- Case 2: $|M| < n/k^6$

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$
  - Call the algorithm recursively on G/M
  - "Uncontract" the k-lean tree decomposition of G/M to get a tree decomposition of G with adhesion size $< 2k$ and $(2k, k)$-unbreakable bags
  - Apply the improver algorithm and return
- Case 2: $|M| < n/k^6$
  - Find a set X of $|X| = n/4$ I_k-simplicial vertices with degree $\leq 4k$

Generalized Bodlaender's compression

Lemma

If there is improver algorithm with running time $f(k) \cdot m$, then there is an algorithm that in time $k^{\mathcal{O}(1)} \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof: Recursive algorithm by using the improver algorithm:

- Run the sparsifier of [Nagamochi, Ibaraki '92] to ensure $m \leq kn$
- Let M be a maximal matching
- Case 1: $|M| \geq n/k^6$
  - Call the algorithm recursively on G/M
  - "Uncontract" the k-lean tree decomposition of G/M to get a tree decomposition of G with adhesion size $< 2k$ and $(2k, k)$-unbreakable bags
  - Apply the improver algorithm and return
- Case 2: $|M| < n/k^6$
  - Find a set X of $|X| = n/4$ I_k-simplicial vertices with degree $\leq 4k$
  - Eliminate X, call the algorithm recursively, add X back, resulting in (k, k)-unbreakable tree decomposition with adhesion size $< k$, apply the improver algorithm, and return

Part 2: The improver algorithm

Part 2: The improver algorithm

Input: Tree decomposition with

- Adhesion size $< 2k$
- $(2k, k)$-unbreakable bags

Output: k-lean tree decomposition

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Def: Separation (A, B) is **doubly well-linked** if the set $A \cap B$ is well-linked in both $G[A]$ and in $G[B]$

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Def: Separation (A, B) is **doubly well-linked** if the set $A \cap B$ is well-linked in both $G[A]$ and in $G[B]$

Set $X \subseteq V(G)$ is **well-linked** in G if for all separations (A, B) of G, it holds that $|(X \cap A) \cup (A \cap B)| \geq |A \cap B|$ or $|(X \cap B) \cup (A \cap B)| \geq |A \cap B|$.

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Def: Separation (A, B) is **doubly well-linked** if the set $A \cap B$ is well-linked in both $G[A]$ and in $G[B]$

Set $X \subseteq V(G)$ is **well-linked** in G if for all separations (A, B) of G, it holds that $|(X \cap A) \cup (A \cap B)| \geq |A \cap B|$ or $|(X \cap B) \cup (A \cap B)| \geq |A \cap B|$.

Lemma (Informal)

If (A, B) is a doubly well-linked separation with $|A \cap B| < k$, can greedily decompose along (A, B)

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Def: Separation (A, B) is **doubly well-linked** if the set $A \cap B$ is well-linked in both $G[A]$ and in $G[B]$

Set $X \subseteq V(G)$ is **well-linked** in G if for all separations (A, B) of G, it holds that $|(X \cap A) \cup (A \cap B)| \geq |A \cap B|$ or $|(X \cap B) \cup (A \cap B)| \geq |A \cap B|$.

Lemma (Informal)

If (A, B) is a doubly well-linked separation with $|A \cap B| < k$, can greedily decompose along (A, B)

- All separations between subsets of A or B can be uncrossed with (A, B)

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Def: Separation (A, B) is **doubly well-linked** if the set $A \cap B$ is well-linked in both $G[A]$ and in $G[B]$

Set $X \subseteq V(G)$ is **well-linked** in G if for all separations (A, B) of G, it holds that $|(X \cap A) \cup (A \cap B)| \geq |A \cap B|$ or $|(X \cap B) \cup (A \cap B)| \geq |A \cap B|$.

Lemma (Informal)

If (A, B) is a doubly well-linked separation with $|A \cap B| < k$, can greedily decompose along (A, B)

- All separations between subsets of A or B can be uncrossed with (A, B)
- If (C, D) is doubly well-linked separation in $G \triangleleft (A, B)$, then the corresponding separation of G is doubly well-linked in G

Doubly well-linked separations

- Suppose we have a separation (A, B) of size $|A \cap B| < k$. Can we decompose along (A, B)?

Def: Separation (A, B) is **doubly well-linked** if the set $A \cap B$ is well-linked in both $G[A]$ and in $G[B]$

Set $X \subseteq V(G)$ is **well-linked** in G if for all separations (A, B) of G, it holds that $|(X \cap A) \cup (A \cap B)| \geq |A \cap B|$ or $|(X \cap B) \cup (A \cap B)| \geq |A \cap B|$.

Lemma (Informal)

If (A, B) is a doubly well-linked separation with $|A \cap B| < k$, can greedily decompose along (A, B)

- All separations between subsets of A or B can be uncrossed with (A, B)
- If (C, D) is doubly well-linked separation in $G \triangleleft (A, B)$, then the corresponding separation of G is doubly well-linked in G

Lemma

If there exists separation (A, B) with $|A \cap B| < k$ and $|A|, |B| \geq s \cdot 2^k$, then exists doubly well-linked separation (A', B') with $|A' \cap B'| < k$ and $|A|, |B| \geq s$.

Issue

Issue: These properties of doubly well-linked separations are morally true, but fail subtly

Hypergraphs and Robertson-Seymour style separations

Solution: Different definitions

Hypergraphs and Robertson-Seymour style separations

Solution: Different definitions

- Graphs $\Rightarrow$ hypergraphs

Hypergraphs and Robertson-Seymour style separations

Solution: Different definitions

- Graphs $\Rightarrow$ hypergraphs
- Separations $(A, B) \Rightarrow$ separations $(A, \overline{A})$, where $A \subseteq E(G)$ and $\overline{A} = E(G) \setminus A$

Hypergraphs and Robertson-Seymour style separations

Solution: Different definitions

- Graphs $\Rightarrow$ hypergraphs
- Separations $(A, B) \Rightarrow$ separations $(A, \overline{A})$, where $A \subseteq E(G)$ and $\overline{A} = E(G) \setminus A$
- Tree decompositions $\Rightarrow$ superbranch decompositions

The real goal

The real goal:

The real goal

The real goal:

Compute a superbranch decomposition, where

- Separations corresponding to adhesions have size $< k$
- Separations corresponding to adhesions are doubly well-linked
- Each torso is $(2^{\mathcal{O}(k)}, k)$-unbreakable

The real goal

The real goal:

Compute a superbranch decomposition, where

- Separations corresponding to adhesions have size $< k$
- Separations corresponding to adhesions are doubly well-linked
- Each torso is $(2^{\mathcal{O}(k)}, k)$-unbreakable

Then,

The real goal

The real goal:

Compute a superbranch decomposition, where

- Separations corresponding to adhesions have size $< k$
- Separations corresponding to adhesions are doubly well-linked
- Each torso is $(2^{\mathcal{O}(k)}, k)$-unbreakable

Then,

- Compute k-lean tree decomposition of each torso (of the primal graph)

The real goal

The real goal:

Compute a superbranch decomposition, where

- Separations corresponding to adhesions have size $< k$
- Separations corresponding to adhesions are doubly well-linked
- Each torso is $(2^{\mathcal{O}(k)}, k)$-unbreakable

Then,

- Compute k-lean tree decomposition of each torso (of the primal graph)
- Combine along the decomposition to get k-lean tree decomposition of the graph

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Refining a decomposition:

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Refining a decomposition:

1. Downwards well-linked

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Refining a decomposition:

1. Downwards well-linked
2. Upwards k-well-linked

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Refining a decomposition:

1. Downwards well-linked
2. Upwards k-well-linked
3. k-tangle-unbreakable torsos

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Refining a decomposition:

1. Downwards well-linked
2. Upwards k-well-linked
3. k-tangle-unbreakable torsos
4. Small adhesions

Steps

Input: Superbranch decomposition with

- Adhesions of size $< 2k$
- $(2k, k)$-unbreakable bags

Goal: Superbranch decomposition with

- Doubly well-linked adhesions of size $< k$
- $(2^{\mathcal{O}(k)}, k)$-unbreakable torsos

Refining a decomposition:

1. Downwards well-linked
2. Upwards k-well-linked
3. k-tangle-unbreakable torsos
4. Small adhesions
5. From k-tangle-unbreakability to $(2^{\mathcal{O}(k)}, k)$-unbreakability

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

$\Rightarrow$ Application: Unbreakable decomposition in linear FPT time

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

$\Rightarrow$ Application: Unbreakable decomposition in linear FPT time

$\Rightarrow$ Application: k-edge-connected components in linear-time (long-standing open problem)

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

$\Rightarrow$ Application: Unbreakable decomposition in linear FPT time

$\Rightarrow$ Application: k-edge-connected components in linear-time (long-standing open problem)

Main techniques:

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

⇒ Application: Unbreakable decomposition in linear FPT time

⇒ Application: k-edge-connected components in linear-time (long-standing open problem)

Main techniques:

- Generalized Bodlaender's compression scheme

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

$\Rightarrow$ Application: Unbreakable decomposition in linear FPT time

$\Rightarrow$ Application: k-edge-connected components in linear-time (long-standing open problem)

Main techniques:

- Generalized Bodlaender's compression scheme
- Decomposition by doubly well-linked separations

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition

$\Rightarrow$ Application: Unbreakable decomposition in linear FPT time

$\Rightarrow$ Application: k-edge-connected components in linear-time (long-standing open problem)

Main techniques:

- Generalized Bodlaender's compression scheme
- Decomposition by doubly well-linked separations

Thank you!