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

Tuukka Korhonen

BWAG ’25

30 October 2025

k-Edge-Connected Components

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

Obs: This gives an equivalence relation among vertices

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])
- $\mathcal{O}(m)$ for $k = 4$ [Nadara, Radecki, Smulewicz, Sokolowski'21, Georgiadis, Italiano, Kosinas'21]

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])
- $\mathcal{O}(m)$ 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 are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])
- $\mathcal{O}(m)$ 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 are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])
- $\mathcal{O}(m)$ 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]
- $m^{1+o(1)}$ for all k [Abboud, Li, Panigrahi, Saranurak '23]

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])
- $\mathcal{O}(m)$ 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]
- $m^{1+o(1)}$ for all k [Abboud, Li, Panigrahi, Saranurak '23]

For minimum cut:

k-Edge-Connected Components

Def: Vertices u and v are k-edge-connected if no (u, v)-cut with $< k$ edges

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

Theorem (This work)

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

Previous results:

- $\mathcal{O}(m)$ for $k = 2$ [Tarjan '72]
- $\mathcal{O}(m)$ for $k = 3$ [Galil & Italiano '91] (using [Hopcroft & Tarjan '73])
- $\mathcal{O}(m)$ 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]
- $m^{1+o(1)}$ for all k [Abboud, Li, Panigrahi, Saranurak '23]

For minimum cut:

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

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)}m$ time
  - Previously $k^{\mathcal{O}(1)}m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)}m$ time

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \operatorname{polylog} m)$ [Forster, Nanongkai, Yang, Saranurak, Yingchareonthawornchai '20]

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \operatorname{polylog} m)$ [Forster, Nanongkai, Yang, Saranurak, Yingchareonthawornchai '20]
- Element connectivity k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \operatorname{polylog} m)$ [Forster, Nanongkai, Yang, Saranurak, Yingchareonthawornchai '20]
- Element connectivity k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k \cdot m^{1+o(1)}$ [Pettie, Saranurak, Yin '22]

k-Lean Tree Decompositions and More

Main technical result:

Theorem (This work)

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

Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \operatorname{polylog} m)$ [Forster, Nanongkai, Yang, Saranurak, Yingchareonthawornchai '20]
- Element connectivity k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k \cdot m^{1+o(1)}$ [Pettie, Saranurak, Yin '22]
- k-Unbreakable tree decomposition in $k^{\mathcal{O}(k^2)} m$ time (with optimal unbreakability parameters)

k-Lean Tree Decompositions and More

Main technical result:

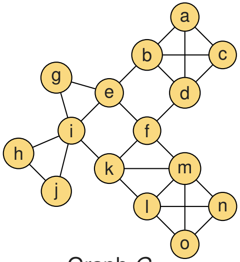
Theorem (This work)

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

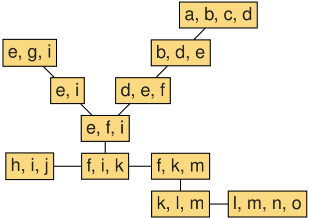
Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k^{\mathcal{O}(1)} m \operatorname{polylog} m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \operatorname{polylog} m)$ [Forster, Nanongkai, Yang, Saranurak, Yingchareonthawornchai '20]
- Element connectivity k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $k \cdot m^{1+o(1)}$ [Pettie, Saranurak, Yin '22]
- k-Unbreakable tree decomposition in $k^{\mathcal{O}(k^2)} m$ time (with optimal unbreakability parameters)
  - Previously $k^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ [Cygan, Komosa, Lokshtanov, Pilipczuk, Pilipczuk, Saurabh, Wahlström '21]
  - and $k^{\mathcal{O}(k)} m^{1+o(1)}$ [Anand, Lee, Li, Long, Saranurak '24] (suboptimal unbreakability parameters)

k-Lean Tree Decompositions

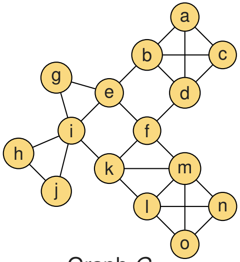


Graph G

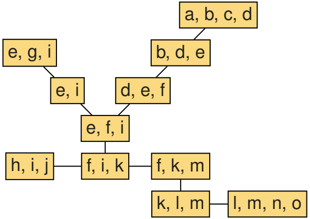


A 3-lean tree decomposition of G

k-Lean Tree Decompositions



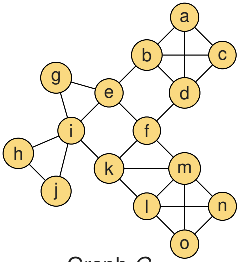
Graph G



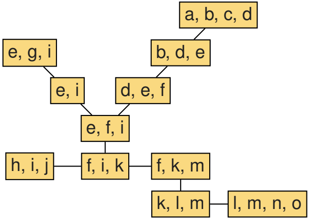
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree

k-Lean Tree Decompositions



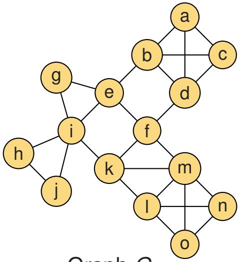
Graph G



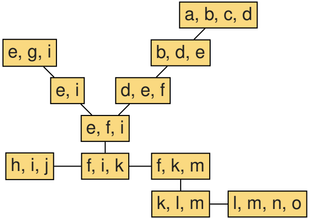
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:

k-Lean Tree Decompositions



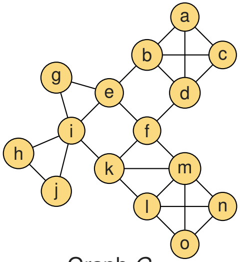
Graph G



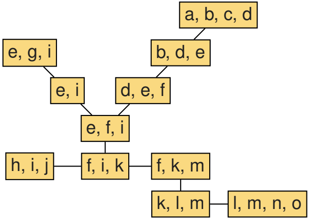
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$

k-Lean Tree Decompositions



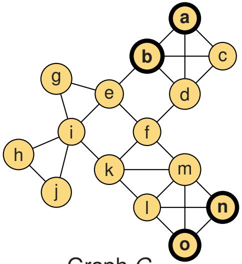
Graph G



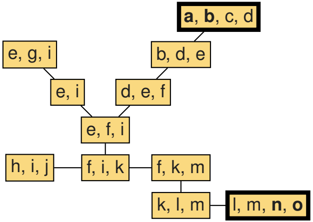
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



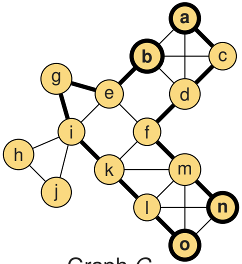
Graph G



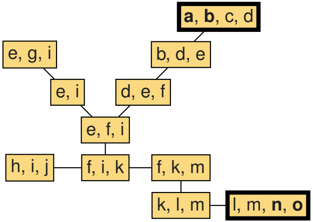
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



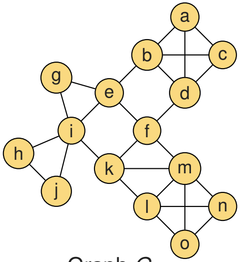
Graph G



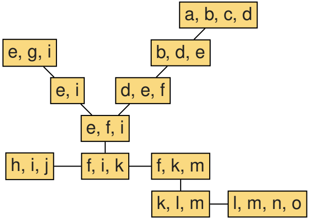
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



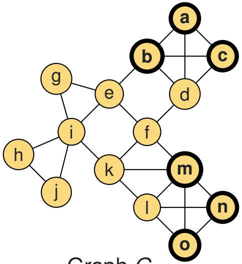
Graph G



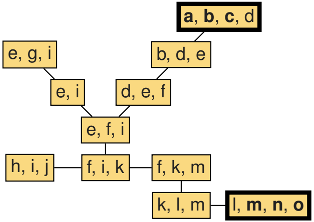
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



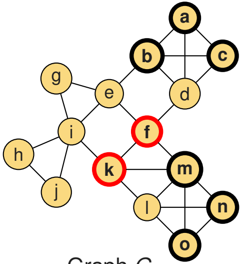
Graph G



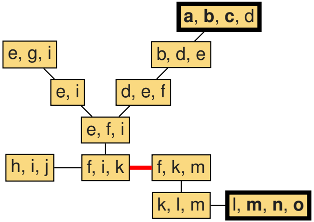
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



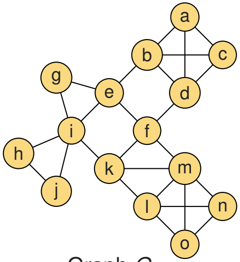
Graph G



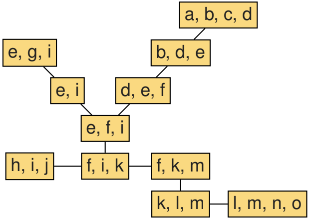
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



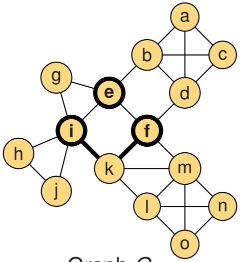
Graph G



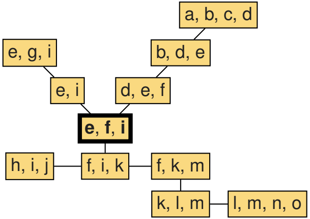
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$

k-Lean Tree Decompositions



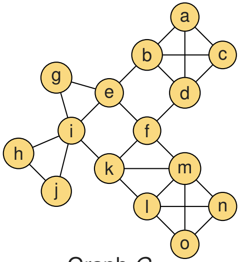
Graph G



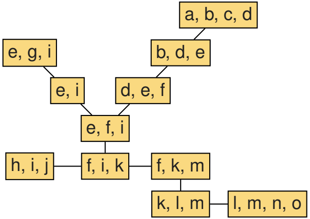
A 3-lean tree decomposition of G

- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$
  - Holds also when $X_1 = X_2$, e.g. $X_1 = X_2 = \{e, f, i\}$ and $Y_1 = \{e, i\}$, $Y_2 = \{e, f\}$.

k-Lean Tree Decompositions



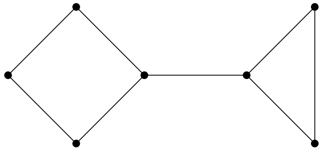
Graph G



A 3-lean tree decomposition of G

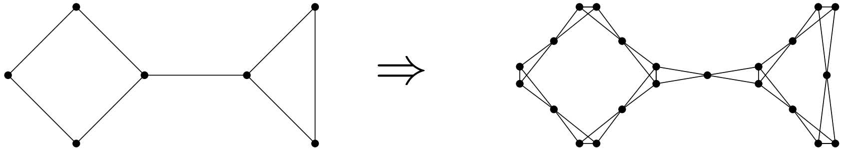
- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For all pairs of bags X_1, X_2 and subsets $Y_1 \subseteq X_1$, $Y_2 \subseteq X_2$ with $|Y_1| = |Y_2| \leq k$, the sets Y_1 and Y_2 can be linked by vertex-disjoint paths if and only if there is no (X_1, X_2)-adhesion of size $< |Y_1| = |Y_2|$
  - Holds also when $X_1 = X_2$, e.g. $X_1 = X_2 = \{e, f, i\}$ and $Y_1 = \{e, i\}$, $Y_2 = \{e, f\}$.
- Defined by [Thomas '90] (for $k = \infty$), and [Carmesin, Diestel, Hamann, and Hundertmark '14]

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



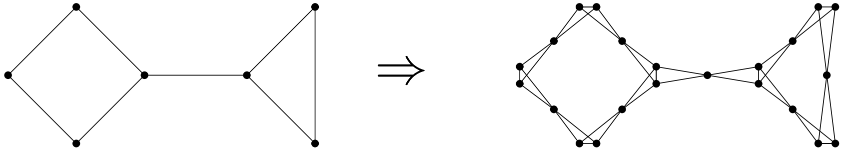
Reducing k-edge-connected components to k-lean tree decomposition

- 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 a k-Gomory-Hu tree



The algorithm

The algorithm

Part 1: Proof that an "improver algorithm" implies the algorithm

The algorithm

Part 1: Proof that an "improver algorithm" implies the algorithm (Inspired by [Bodlaender '96])

The algorithm

Part 1: Proof that an "improver algorithm" implies the algorithm (Inspired by [Bodlaender '96])

Part 2: The improver algorithm

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: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

Output: k-lean tree decomposition

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

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.

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

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.

Proof idea:

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

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.

Proof idea:

- Can sparsify to $m = \mathcal{O}(kn)$ by [Nagamochi & Ibaraki '92]

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

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.

Proof idea:

- Can sparsify to $m = \mathcal{O}(kn)$ by [Nagamochi & Ibaraki '92]

Case 1: Matching M of size $\Omega(n/\text{poly}(k)) \Rightarrow$ contract M, recurse, uncontract, and apply improver algorithm

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

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.

Proof idea:

- Can sparsify to $m = \mathcal{O}(kn)$ by [Nagamochi & Ibaraki '92]

Case 1: Matching M of size $\Omega(n/\text{poly}(k)) \Rightarrow$ contract M, recurse, uncontract, and apply improver algorithm

Case 2: No matching of size $\Omega(n/\text{poly}(k)) \Rightarrow$ similar recursion in some other way...

Part 2: The improver algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

Output: k-lean tree decomposition

Part 2: The improver algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

Output: k-lean tree decomposition

Lemma

There is an improver algorithm with running time $k^{\mathcal{O}(k^2)} m$.

Part 2: The improver algorithm

Improver algorithm:

Input: $(2k, k)$-unbreakable tree decomposition with adhesion $< 2k$:

- Adhesion $< 2k$: Any two adjacent bags X_u and X_v have $|X_u \cap X_v| < 2k$
- $(2k, k)$-unbreakable: For each bag X, no separation (A, B) of G of order $|A \cap B| < k$ with $\min(|A \cap X|, |B \cap X|) \geq 2k$

Output: k-lean tree decomposition

Lemma

There is an improver algorithm with running time $k^{\mathcal{O}(k^2)} m$.

Idea:

- Slowly improve the properties of the input decomposition (in 6 consecutive steps)

The improver algorithm

Goal

Want to compute a superbranch decomposition T of G so that

- T has adhesion size $< k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable, and
- internal separations of T are doubly well-linked

Goal

Want to compute a superbranch decomposition T of G so that

- T has adhesion size $< k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable, and
- internal separations of T are doubly well-linked

Lemma

Such T can be turned into a k-lean tree decomposition by replacing each torso by a k-lean tree decomposition of its primal graph.

Goal

Want to compute a superbranch decomposition T of G so that

- T has adhesion size $< k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable, and
- internal separations of T are doubly well-linked

Lemma

Such T can be turned into a k-lean tree decomposition by replacing each torso by a k-lean tree decomposition of its primal graph.

Lemma

A k-lean tree decomposition of an (s, k)-unbreakable graph can be computed in $s^{\mathcal{O}(k)} m$ time.

Focus

$2^{\mathcal{O}(k^2)}m$-time algorithm for a problem with:

Input: Graph G and superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$, and
- torsos of T are $(2k, k)$-unbreakable in G.

Output: Superbranch decomposition T of G s.t.

- T has adhesion size $< k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable, and
- internal separations of T are doubly well-linked.

Step 1: Downwards well-linkedness

Input: Graph G and superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$, and
- torsos of T are $(2k, k)$-unbreakable in G.

Output: Superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable in G, and
- T is downwards well-linked.

Step 2: Upwards k-well-linkedness

Input: Graph G and a superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable in G, and
- T is downwards well-linked.

Output: Superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable,
- T is downwards well-linked, and
- T is upwards k-well-linked.

Step 3: Tangle k-unbreakability

Input: Graph G and a superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$,
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable,
- T is downwards well-linked, and
- T is upwards k-well-linked.

Output: Superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$,
- T is downwards well-linked,
- T is upwards k-well-linked, and
- torsos of T are k-tangle-unbreakable.

Step 4: Small adhesions

Input: Graph G and a superbranch decomposition T of G s.t.

- T has adhesion size $< 2k$,
- T is downwards well-linked,
- T is upwards k-well-linked, and
- torsos of T are k-tangle-unbreakable.

Output: Superbranch decomposition T of G s.t.

- T has adhesion size $< k$,
- internal separations of T are doubly well-linked, and
- torsos of T are k-tangle-unbreakable.

Step 5: Unbreakable torsos

Input: Graph G and a superbranch decomposition T of G s.t.

- T has adhesion size $< k$,
- internal separations of T are doubly well-linked, and
- torsos of T are k-tangle-unbreakable.

Output: Superbranch decomposition T of G s.t.

- T has adhesion size $< k$,
- internal separations of T are doubly well-linked, and
- torsos of T are $(2^{\mathcal{O}(k)}, k)$-unbreakable.

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
- Implies $k^{\mathcal{O}(k^2)}m$ time algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition...

Conclusion

- $k^{\mathcal{O}(k^2)}m$ time algorithm for k-lean tree decomposition
- Implies $k^{\mathcal{O}(k^2)}m$ time algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition...

Techniques:

- Recursive matching contraction compression (inspired by [Bodlaender'93])
- Decomposition by doubly well-linked separations

Conclusion

- $k^{\mathcal{O}(k^2)}m$ time algorithm for k-lean tree decomposition
- Implies $k^{\mathcal{O}(k^2)}m$ time algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition...

Techniques:

- Recursive matching contraction compression (inspired by [Bodlaender'93])
- Decomposition by doubly well-linked separations

Follow-up work:

- Dynamic treewidth in logarithmic time

Conclusion

- $k^{\mathcal{O}(k^2)}m$ time algorithm for k-lean tree decomposition
- Implies $k^{\mathcal{O}(k^2)}m$ time algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition...

Techniques:

- Recursive matching contraction compression (inspired by [Bodlaender'93])
- Decomposition by doubly well-linked separations

Follow-up work:

- Dynamic treewidth in logarithmic time

Open problems/future work:

- $\text{poly}(k)m$ time for k-edge-connected components?
- Simpler algorithm?
- Tutte-like decomposition in linear-time?

Conclusion

- $k^{\mathcal{O}(k^2)}m$ time algorithm for k-lean tree decomposition
- Implies $k^{\mathcal{O}(k^2)}m$ time algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition...

Techniques:

- Recursive matching contraction compression (inspired by [Bodlaender'93])
- Decomposition by doubly well-linked separations

Follow-up work:

- Dynamic treewidth in logarithmic time

Open problems/future work:

- $\text{poly}(k)m$ time for k-edge-connected components?
- Simpler algorithm?
- Tutte-like decomposition in linear-time?

Thank you!