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

Tuukka Korhonen

STOC 2025

23 June 2025

Graph connectivity problems

Almost-linear-time algorithms for many problems:

- Edge connectivity (global edge min-cut), $\mathcal{O}(m \log^3 n)$ [Karger '96]
- Vertex connectivity (global vertex min-cut), $\mathcal{O}(m^{1+o(1)})$ [Li, Nanongkai, Panigrahi, Saranurak & Yingchareonthawornchai '21]
- Gomory-Hu tree, $\mathcal{O}(m^{1+o(1)})$ [Abboud, Li, Panigrahi & Saranurak '23]

Graph connectivity problems

Almost-linear-time algorithms for many problems:

- Edge connectivity (global edge min-cut), $\mathcal{O}(m \log^3 n)$ [Karger '96]
- Vertex connectivity (global vertex min-cut), $\mathcal{O}(m^{1+o(1)})$ [Li, Nanongkai, Panigrahi, Saranurak & Yingchareonthawornchai '21]
- Gomory-Hu tree, $\mathcal{O}(m^{1+o(1)})$ [Abboud, Li, Panigrahi & Saranurak '23]

No idea about truly linear-time!

Graph connectivity problems

Almost-linear-time algorithms for many problems:

- Edge connectivity (global edge min-cut), $\mathcal{O}(m \log^3 n)$ [Karger '96]
- Vertex connectivity (global vertex min-cut), $\mathcal{O}(m^{1+o(1)})$ [Li, Nanongkai, Panigrahi, Saranurak & Yingchareonthawornchai '21]
- Gomory-Hu tree, $\mathcal{O}(m^{1+o(1)})$ [Abboud, Li, Panigrahi & Saranurak '23]

No idea about truly linear-time!

This work

$f(k) \cdot m$ time algorithms for connectivity problems restricted to cuts/separators of size $< k$

Graph connectivity problems

Almost-linear-time algorithms for many problems:

- Edge connectivity (global edge min-cut), $\mathcal{O}(m \log^3 n)$ [Karger '96]
- Vertex connectivity (global vertex min-cut), $\mathcal{O}(m^{1+o(1)})$ [Li, Nanongkai, Panigrahi, Saranurak & Yingchareonthawornchai '21]
- Gomory-Hu tree, $\mathcal{O}(m^{1+o(1)})$ [Abboud, Li, Panigrahi & Saranurak '23]

No idea about truly linear-time!

This work

$f(k) \cdot m$ time algorithms for connectivity problems restricted to cuts/separators of size $< k$

- Deterministic!

Graph connectivity problems

Almost-linear-time algorithms for many problems:

- Edge connectivity (global edge min-cut), $\mathcal{O}(m \log^3 n)$ [Karger '96]
- Vertex connectivity (global vertex min-cut), $\mathcal{O}(m^{1+o(1)})$ [Li, Nanongkai, Panigrahi, Saranurak & Yingchareonthawornchai '21]
- Gomory-Hu tree, $\mathcal{O}(m^{1+o(1)})$ [Abboud, Li, Panigrahi & Saranurak '23]

No idea about truly linear-time!

This work

$f(k) \cdot m$ time algorithms for connectivity problems restricted to cuts/separators of size $< k$

- Deterministic!
- One graph decomposition to rule them all!

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 an 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 an 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 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 in the same k-edge-connected component 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 in the same k-edge-connected component 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&Sokołowski'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 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&Sokołowski'21, Georgiadis,Italiano&Kosinas'21]
- $\mathcal{O}(m)$ for $k = 5$ [Kosinas '24]
- $\text{poly}(k) \cdot m \operatorname{polylog} m$ 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 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&Sokołowski'21, Georgiadis,Italiano&Kosinas'21]
- $\mathcal{O}(m)$ for $k = 5$ [Kosinas '24]
- $\text{poly}(k) \cdot m \,\text{polylog}\, m$ 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 in the same k-edge-connected component 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&Sokołowski'21, Georgiadis,Italiano&Kosinas'21]
- $\mathcal{O}(m)$ for $k = 5$ [Kosinas '24]
- $\text{poly}(k) \cdot m \text{ polylog } m$ 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], $\mathcal{O}(m \text{ polylog } m)$ [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 $\text{poly}(k) \cdot m \, \text{polylog} \, m$ [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 $\text{poly}(k) \cdot m \,\text{polylog}\, m$ [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 $\text{poly}(k) \cdot m \, \text{polylog} \, m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \, \text{polylog} \, m)$ [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 $\mathrm{poly}(k) \cdot m \,\mathrm{polylog}\, m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \,\mathrm{polylog}\, m)$ [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 $\text{poly}(k) \cdot m \text{ polylog } m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \text{ polylog } m)$ [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 $\text{poly}(k) \cdot m \,\text{polylog}\, m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \,\text{polylog}\, m)$ [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:

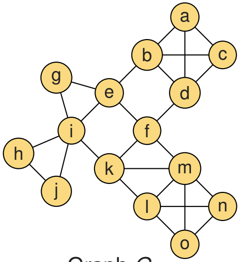
Theorem (This work)

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

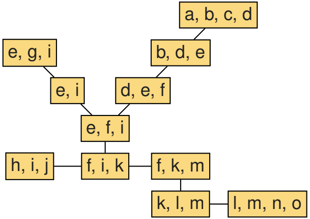
Implies the first "parameterized linear-time" ($f(k) \cdot m$ time) algorithms for many problems:

- k-Gomory-Hu tree in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\text{poly}(k) \cdot m \, \text{polylog} \, m$ [Hariharan, Kavitha, Panigrahi '07]
- k-Vertex connectivity in $k^{\mathcal{O}(k^2)} m$ time
  - Previously $\mathcal{O}(k^3 m \, \text{polylog} \, m)$ [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

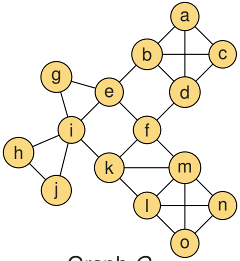


Graph G

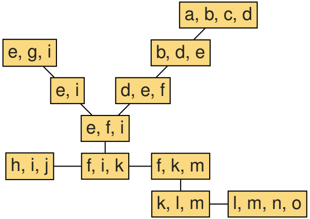


A 3-lean tree decomposition of G

k-Lean Tree Decompositions



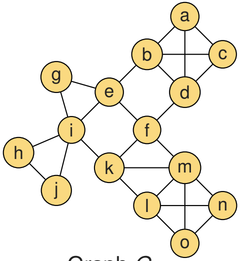
Graph G



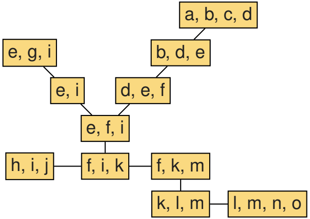
A 3-lean tree decomposition of G

- 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



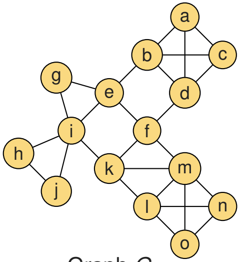
Graph G



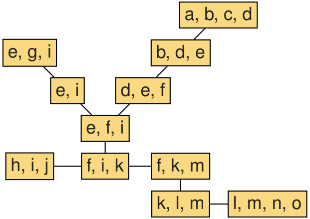
A 3-lean tree decomposition of G

- 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



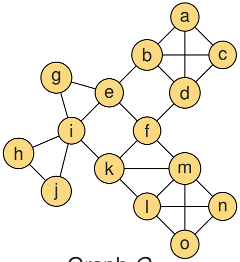
Graph G



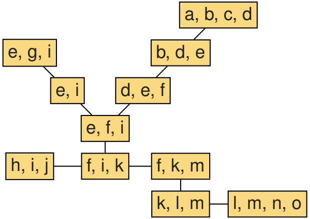
A 3-lean tree decomposition of G

- 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



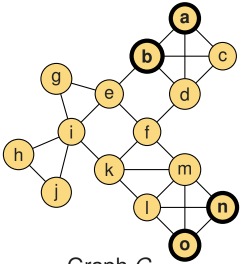
Graph G



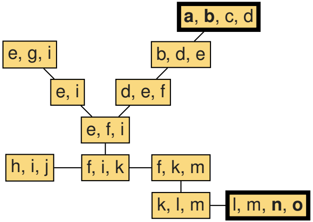
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



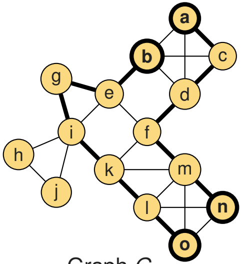
Graph G



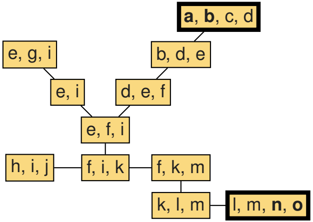
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



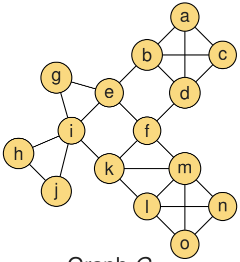
Graph G



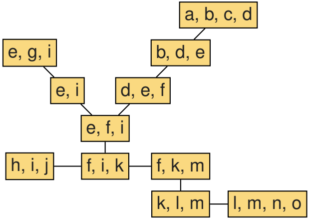
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



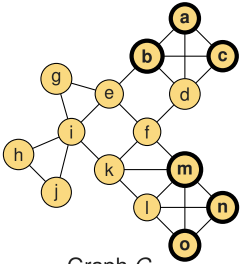
Graph G



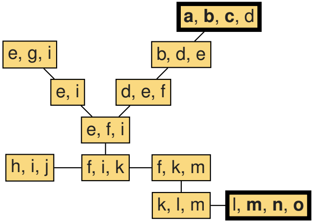
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



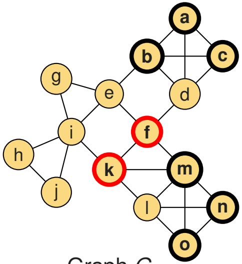
Graph G



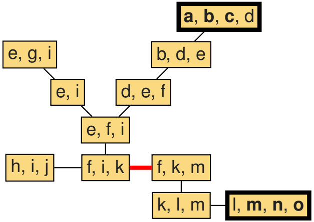
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



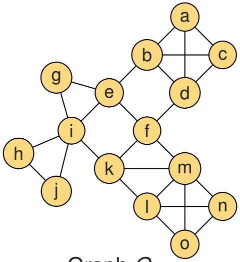
Graph G



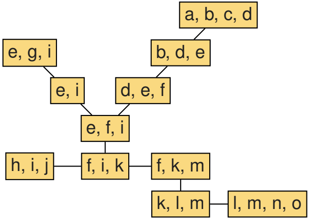
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



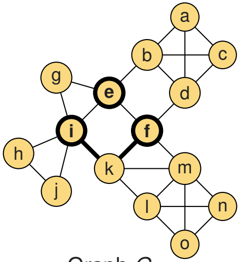
Graph G



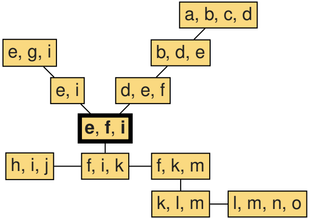
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$

k-Lean Tree Decompositions



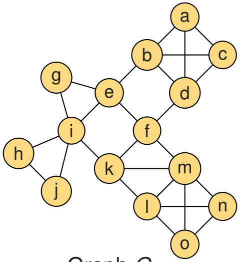
Graph G



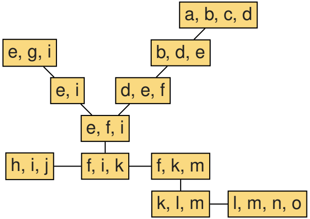
A 3-lean tree decomposition of G

- 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 each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$
  - Holds also when $B_1 = B_2$, e.g. $B_1 = B_2 = \{e, f, i\}$ and $X_1 = \{e, i\}$, $X_2 = \{e, f\}$.

k-Lean Tree Decompositions



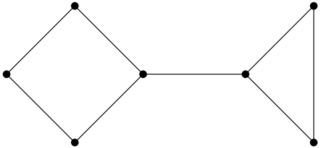
Graph G



A 3-lean tree decomposition of G

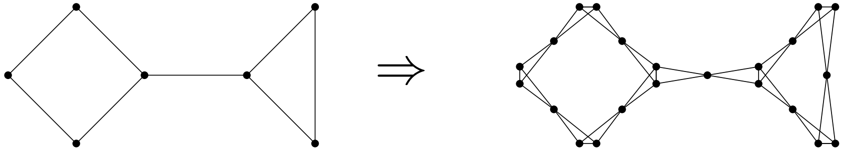
- Tree decomposition:
  1. All vertices and edges are covered by bags
  2. For each vertex v, the bags containing v form a connected subtree
- k-lean:
  1. The adhesions (i.e. intersections of adjacent bags) have size $< k$
  2. For each pair of bags B_1, B_2 and subsets $X_1 \subseteq B_1$, $X_2 \subseteq B_2$ with $|X_1| = |X_2| \leq k$, the sets X_1 and X_2 can be linked by vertex-disjoint paths iff there is no (B_1, B_2)-adhesion of size $< |X_1| = |X_2|$
  - Holds also when $B_1 = B_2$, e.g. $B_1 = B_2 = \{e, f, i\}$ and $X_1 = \{e, i\}$, $X_2 = \{e, f\}$.
- Defined by [Thomas '90] (for $k = \infty$), and [Carmesin, Diestel, Hamann, and Hundertmark '14]

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



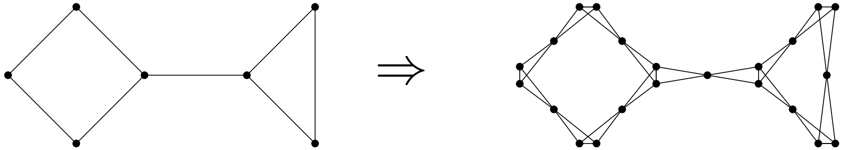
Reducing k-edge-connected components to k-lean tree decomposition

- 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 “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: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

Output: k-lean tree decomposition

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

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 $\text{poly}(k) \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

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 $\text{poly}(k) \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof idea:

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

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 $\text{poly}(k) \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof idea:

- Can always assume $m = \mathcal{O}(kn)$ by [Nagamochi, Ibaraki '92]

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

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 $\text{poly}(k) \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof idea:

- Can always assume $m = \mathcal{O}(kn)$ by [Nagamochi, Ibaraki '92]

Case 1: Matching M of size $\Omega(n) \Rightarrow$ contract M, recurse, uncontract, and apply improver algorithm

Part 1: Improver algorithm implies the algorithm

Improver algorithm:

Input: A “weakly-k-lean” tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

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 $\text{poly}(k) \cdot f(k) \cdot m$ computes a k-lean tree decomposition.

Proof idea:

- Can always assume $m = \mathcal{O}(kn)$ by [Nagamochi, Ibaraki ’92]

Case 1: Matching M of size $\Omega(n) \Rightarrow$ contract M, recurse, uncontract, and apply improver algorithm

Case 2: No matching of size $\Omega(n) \Rightarrow$ manage to recurse in some other way...

Part 2: The improver algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

Output: k-lean tree decomposition

Part 2: The improver algorithm

Improver algorithm:

Input: A “weakly-k-lean” tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

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: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

Output: k-lean tree decomposition

Lemma

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

Proof idea:

Part 2: The improver algorithm

Improver algorithm:

Input: A "weakly-k-lean" tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

Output: k-lean tree decomposition

Lemma

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

Proof idea:

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

Part 2: The improver algorithm

Improver algorithm:

Input: A “weakly-k-lean” tree decomposition:

- Adhesion size $< 2k$
- Any two subsets $X_1, X_2 \subseteq B$ of a bag B of size $|X_1|, |X_2| \geq 2k$ can be linked by k vertex-disjoint paths

Output: k-lean tree decomposition

Lemma

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

Proof idea:

- Slowly improve the properties of the input tree decomposition (in 6 consecutive steps)
- Key tool: Decomposition by doubly well-linked separations

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition, implying algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition
  - k-Gomory-Hu tree (for both edge- and element-connectivity)

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition, implying algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition
  - k-Gomory-Hu tree (for both edge- and element-connectivity)

Main techniques:

- Recursive matching contraction compression (inspired by [Bodlaender '93])
- Decomposition by doubly well-linked separations (inspired by [Graph Minors X., Robertson & Seymour '91])

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition, implying algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition
  - k-Gomory-Hu tree (for both edge- and element-connectivity)

Main techniques:

- Recursive matching contraction compression (inspired by [Bodlaender '93])
- Decomposition by doubly well-linked separations (inspired by [Graph Minors X., Robertson & Seymour '91])

Open problems:

- Improve running time to $2^{o(k^2)} m$

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition, implying algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition
  - k-Gomory-Hu tree (for both edge- and element-connectivity)

Main techniques:

- Recursive matching contraction compression (inspired by [Bodlaender '93])
- Decomposition by doubly well-linked separations (inspired by [Graph Minors X., Robertson & Seymour '91])

Open problems:

- Improve running time to $2^{o(k^2)} m$ (even e.g. just for k-edge-connected components)

Conclusion

- $k^{\mathcal{O}(k^2)}m$ time algorithm for k-lean tree decomposition, implying algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition
  - k-Gomory-Hu tree (for both edge- and element-connectivity)

Main techniques:

- Recursive matching contraction compression (inspired by [Bodlaender '93])
- Decomposition by doubly well-linked separations (inspired by [Graph Minors X., Robertson & Seymour '91])

Open problems:

- Improve running time to $2^{o(k^2)}m$ (even e.g. just for k-edge-connected components)
- Simplify

Conclusion

- $k^{\mathcal{O}(k^2)} m$ time algorithm for k-lean tree decomposition, implying algorithms for:
  - k-edge-connected components (long-standing open problem)
  - k-vertex connectivity
  - k-unbreakable tree decomposition
  - k-Gomory-Hu tree (for both edge- and element-connectivity)

Main techniques:

- Recursive matching contraction compression (inspired by [Bodlaender ’93])
- Decomposition by doubly well-linked separations (inspired by [Graph Minors X., Robertson & Seymour ’91])

Open problems:

- Improve running time to $2^{o(k^2)} m$ (even e.g. just for k-edge-connected components)
- Simplify

Thank you!

Feel free to reach out to me for any questions/comments: https://tuukkakorhonen.com/