Computing Tree Decompositions with Small Independence Number

Tuukka Korhonen

UNIVERSITY OF BERGEN

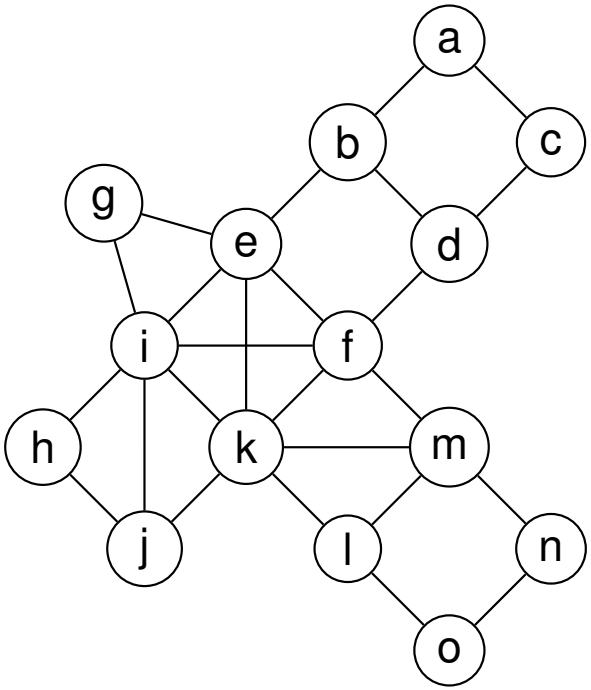
joint work with
Clément Dallard[1], Fedor V. Fomin, Petr A. Golovach, and Martin Milanič[1]

[1] FAMNIT and IAM, University of Primorska

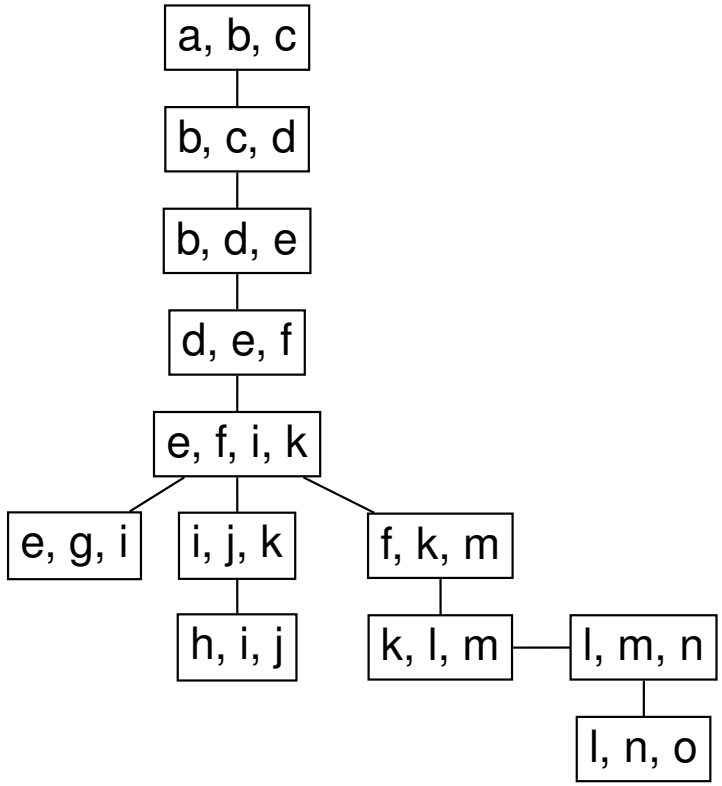
GROW 2022

22 September 2022

Tree Decompositions

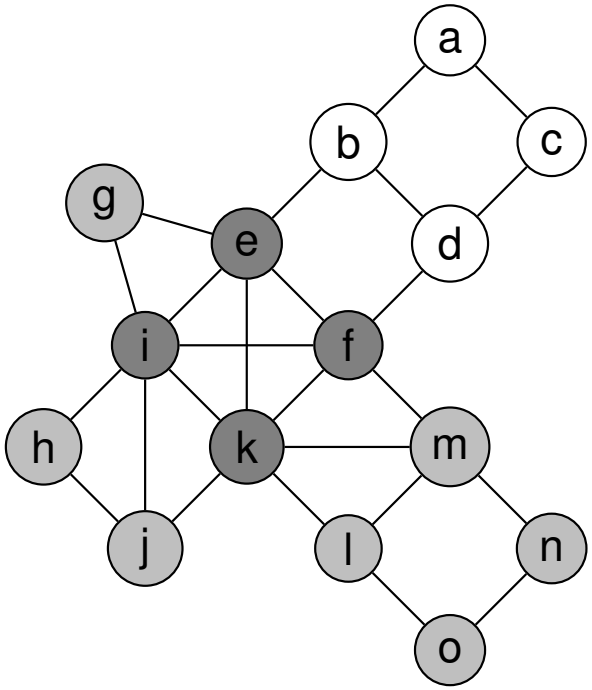


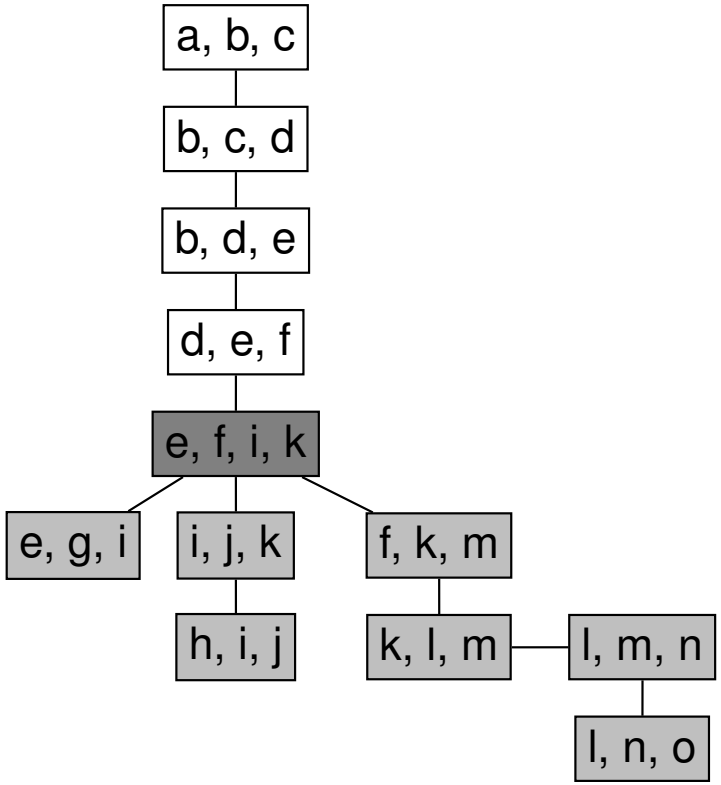
Graph G



A tree decomposition of G

Dynamic programming for maximum independent set

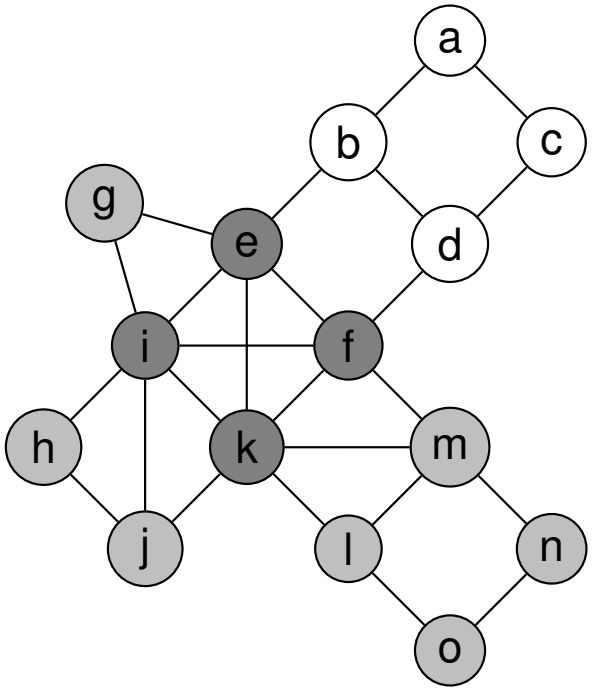


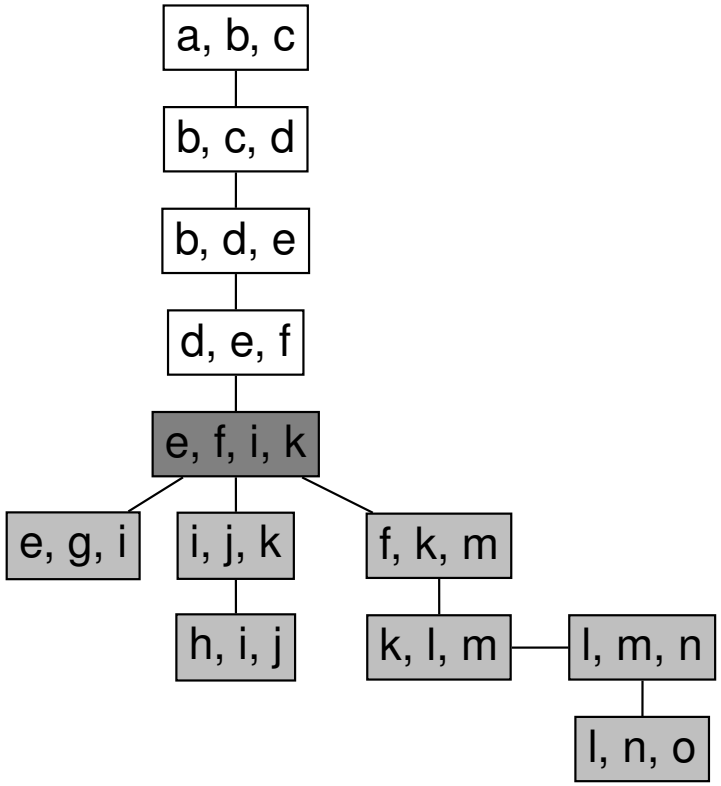


For every node t and subset $S \subseteq B_t$

$\mathrm{dp}[t][S] =$ maximum independent set I below t with $I \cap B_t = S$

Dynamic programming for maximum independent set



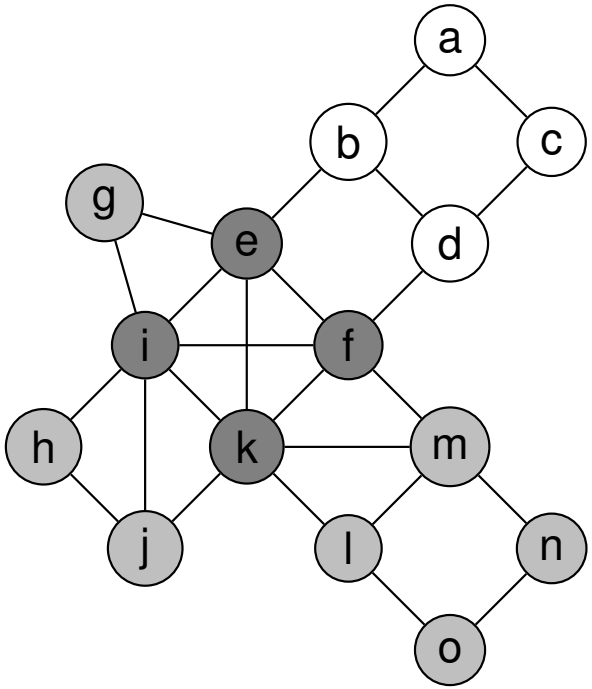


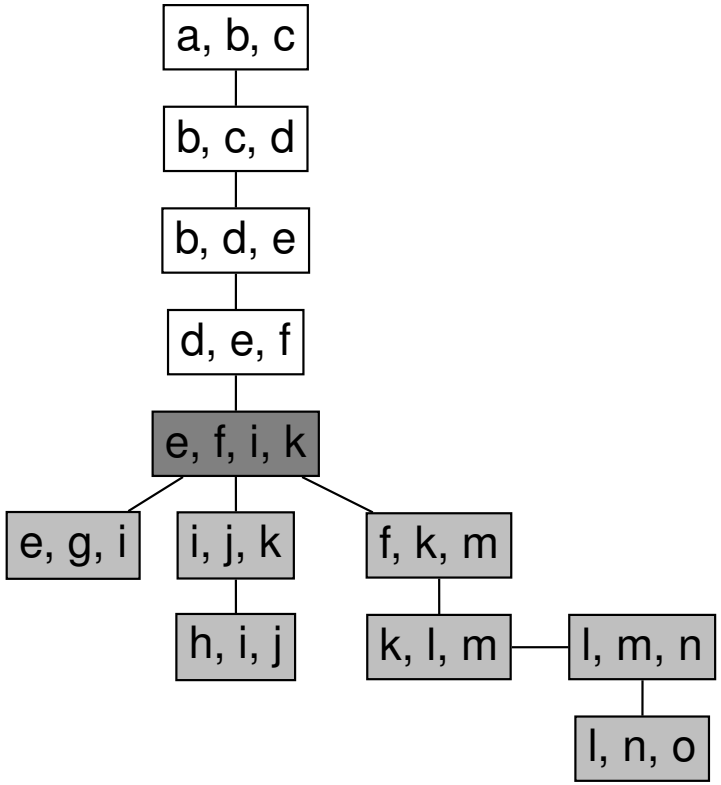
For every node t and subset $S \subseteq B_t$

$\mathrm{dp}[t][S] =$ maximum independent set I below t with $I \cap B_t = S$

$2^{|B_t|}$ states per node

Dynamic programming for maximum independent set





For every node t and **independent** subset $S \subseteq B_t$

$\mathtt{dp}[t][S] =$ maximum independent set I below t with $I \cap B_t = S$

$\#IS(B_t)$ states per node

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

Maximum independent set in B_t has size k: $\#IS(B_t) \leq n^k$

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

Maximum independent set in B_t has size k: $\#IS(B_t) \leq n^k$

The independence number of a tree decomposition: $\alpha(TD) = \max_{B_t} \alpha(B_t)$

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

Maximum independent set in B_t has size k: $\#IS(B_t) \leq n^k$

The independence number of a tree decomposition: $\alpha(TD) = \max_{B_t} \alpha(B_t)$

Tree-independence number: tree-$\alpha(G) = \min_{TD} \alpha(TD)$

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

Maximum independent set in B_t has size k: $\#IS(B_t) \leq n^k$

The independence number of a tree decomposition: $\alpha(TD) = \max_{B_t} \alpha(B_t)$

Tree-independence number: tree-$\alpha(G) = \min_{TD} \alpha(TD)$

- Introduced by [Dallard, Milanič, and Storgel '21]

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

Maximum independent set in B_t has size k: $\#IS(B_t) \leq n^k$

The independence number of a tree decomposition: $\alpha(TD) = \max_{B_t} \alpha(B_t)$

Tree-independence number: tree-$\alpha(G) = \min_{TD} \alpha(TD)$

- Introduced by [Dallard, Milanič, and Storgel '21]
- Most general parameter over tree decompositions that gives XP algorithms for maximum independent set

When can we bound $\#IS(B_t)$?

What kind of tree decompositions have bounded $\#IS(B_t)$?

- Clique-trees of chordal graphs: $\#IS(B_t) \leq n$
- B_t is clique$+k$ vertices: $\#IS(B_t) \leq 2^k n$ [Jacob, Panolan, Raman, Sahlot '20]
- B_t is clique$-k$ edges: $\#IS(B_t) \leq 2^{\sqrt{k}} n$ [Fomin and Golovach '20]
- B_t is covered by k cliques: $\#IS(B_t) \leq n^k$ – used for geometric intersection graphs [De Berg, Bodlaender, Kisfaludi-Bak, Marx, and Van Der Zanden '18]

Maximum independent set in B_t has size k: $\#IS(B_t) \leq n^k$

The independence number of a tree decomposition: $\alpha(TD) = \max_{B_t} \alpha(B_t)$

Tree-independence number: tree-$\alpha(G) = \min_{TD} \alpha(TD)$

- Introduced by [Dallard, Milanič, and Storgel '21]
- Most general parameter over tree decompositions that gives XP algorithms for maximum independent set (with some assumptions)

Algorithmic Applications of Tree-independence number

Let $k = \text{tree-}\alpha(G)$

Algorithmic Applications of Tree-independence number

Let $k = \text{tree-}\alpha(G)$

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set

Algorithmic Applications of Tree-independence number

Let $k = \text{tree-}\alpha(G)$

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set
- $\mathcal{O}(n^{|H|\cdot(k+2)})$ time algorithm for maximum weight H-packing [Dallard, Milanič, and Storgel'21]

Algorithmic Applications of Tree-independence number

Let $k = \text{tree-}\alpha(G)$

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set
- $\mathcal{O}(n^{|H|\cdot(k+2)})$ time algorithm for maximum weight H-packing [Dallard, Milanič, and Storgel'21]
- $n^{\mathcal{O}(k)}$ time algorithms for feedback vertex set, longest induced path, and generalizations [Milanič and Rzazewski'22]

Algorithmic Applications of Tree-independence number

Let $k = \text{tree-}\alpha(G)$

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set
- $\mathcal{O}(n^{|H|\cdot(k+2)})$ time algorithm for maximum weight H-packing [Dallard, Milanič, and Storgel'21]
- $n^{\mathcal{O}(k)}$ time algorithms for feedback vertex set, longest induced path, and generalizations [Milanič and Rzazewski'22]

All applications need the decomposition as an input!

Our Results

Theorem

There is a $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number, which also outputs the corresponding tree decomposition.

Our Results

Theorem

There is a $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number, which also outputs the corresponding tree decomposition.

$\Rightarrow 2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time algorithms for several problems parameterized by tree-independence number k

Our Results

Theorem

There is a $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number, which also outputs the corresponding tree decomposition.

$\Rightarrow 2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time algorithms for several problems parameterized by tree-independence number k

Hardness results:

Our Results

Theorem

There is a $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number, which also outputs the corresponding tree decomposition.

$\Rightarrow 2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time algorithms for several problems parameterized by tree-independence number k

Hardness results:

- Assuming Gap-ETH, no $f(k)n^{o(k)}$ time $g(k)$-approximation algorithm

Our Results

Theorem

There is a $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number, which also outputs the corresponding tree decomposition.

$\Rightarrow 2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time algorithms for several problems parameterized by tree-independence number k

Hardness results:

- Assuming Gap-ETH, no $f(k)n^{o(k)}$ time $g(k)$-approximation algorithm
- For every constant $k \geq 4$, NP-hard to decide if tree-$\alpha(G) \leq k$

Our Results

Theorem

There is a $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number, which also outputs the corresponding tree decomposition.

$\Rightarrow 2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ time algorithms for several problems parameterized by tree-independence number k

Hardness results:

- Assuming Gap-ETH, no $f(k)n^{o(k)}$ time $g(k)$-approximation algorithm
- For every constant $k \geq 4$, NP-hard to decide if tree-$\alpha(G) \leq k$
  - (For $k = 1$ linear time, $k = 2, 3$ remain open)

The Algorithm

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α
- Recursive construction in Robertson-Seymour fashion

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α
- Recursive construction in Robertson-Seymour fashion
  - Reduction from finding balanced separators to finding separators

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α
- Recursive construction in Robertson-Seymour fashion
  - Reduction from finding balanced separators to finding separators
    - 2-approximation algorithm for separators

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α
- Recursive construction in Robertson-Seymour fashion
  - Reduction from finding balanced separators to finding separators
    - 2-approximation algorithm for separators
      1. Container with bounded α

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α
- Recursive construction in Robertson-Seymour fashion
  - Reduction from finding balanced separators to finding separators
    - 2-approximation algorithm for separators
      1. Container with bounded α
      2. Branching

Outline

- Bounded tree-$\alpha \Rightarrow$ balanced separators with bounded α
- Recursive construction in Robertson-Seymour fashion
  - Reduction from finding balanced separators to finding separators
    - 2-approximation algorithm for separators
      1. Container with bounded α
      2. Branching
      3. Linear programming

Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

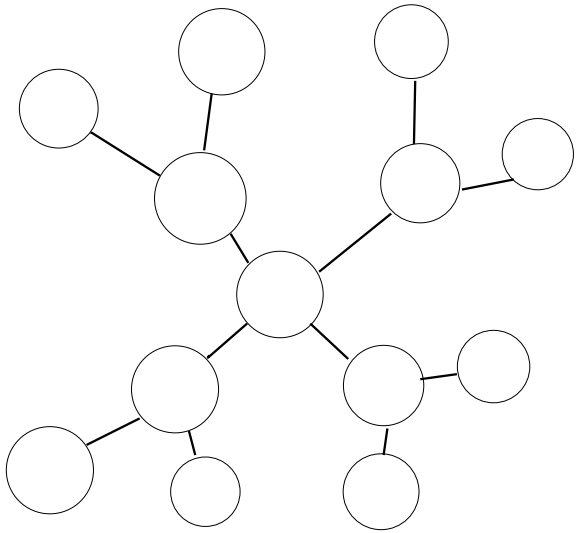
Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:

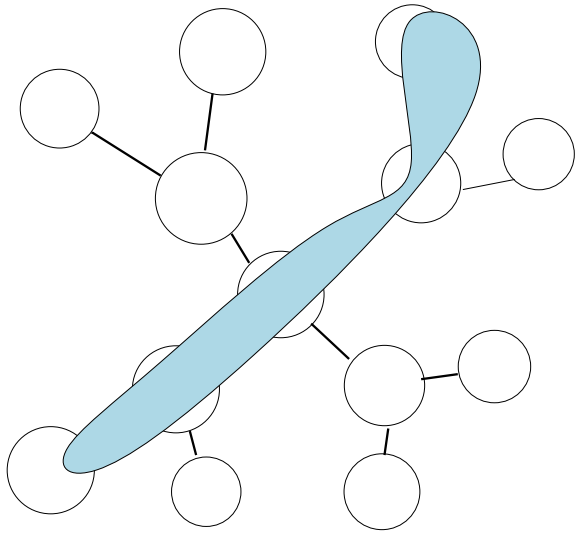


Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:

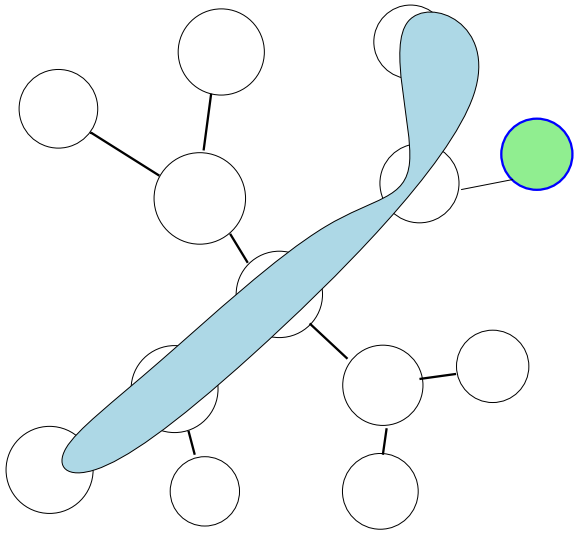


Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:

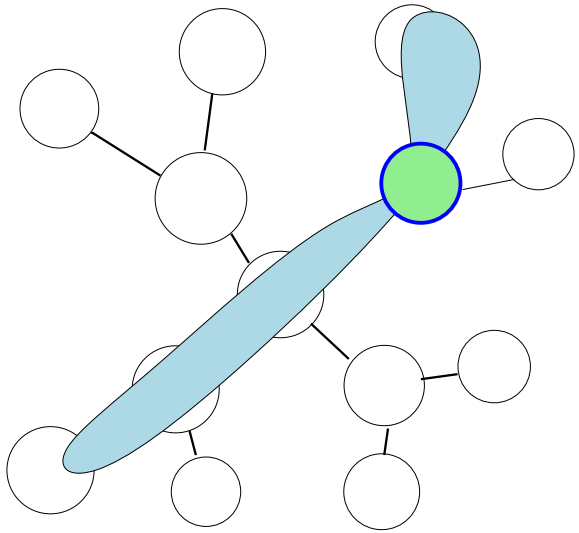


Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:

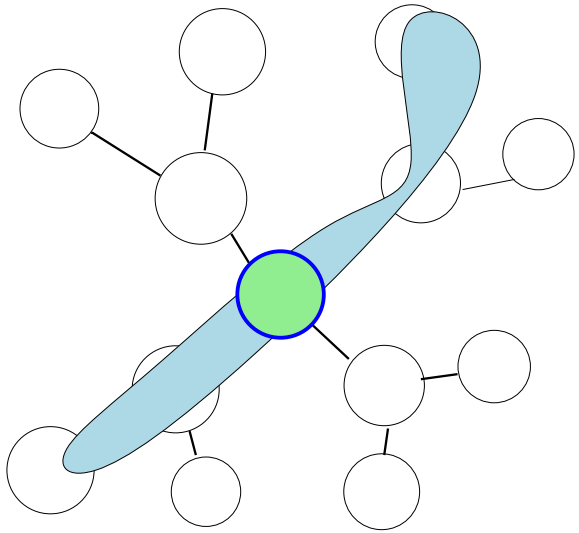


Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:

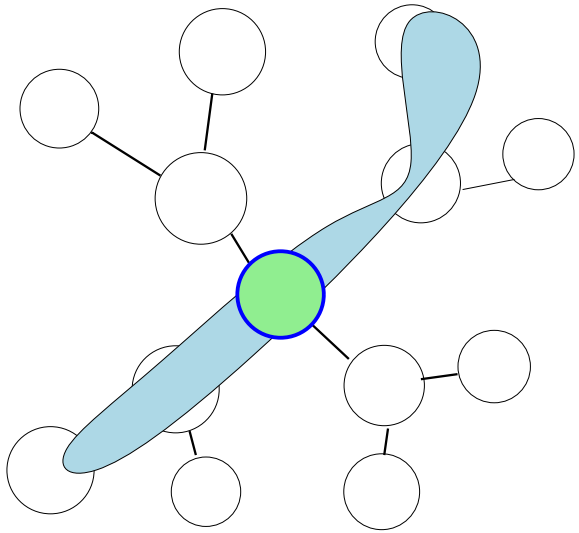


Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:



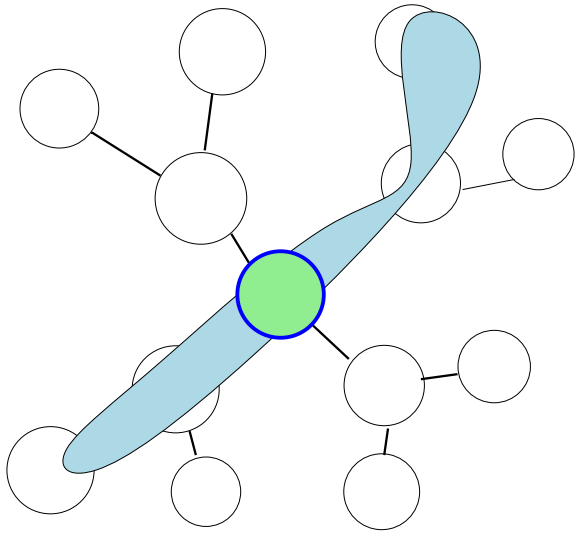
Sufficient to ensure that $\alpha(X \cap C_1) \geq 2k$ and $\alpha(X \cap C_2) \geq 2k$

Balanced separators

Input: Graph G, integer k, and a vertex set X with $\alpha(X) = 9k$

Task: Find a separation (C_1, S, C_2) with $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$ or conclude tree-$\alpha(G) > k$

Why balanced separators *exists*:



Sufficient to ensure that $\alpha(X \cap C_1) \geq 2k$ and $\alpha(X \cap C_2) \geq 2k$

Algorithm: Guess independent set $I_1 \subseteq X \cap C_1$ with $|I_1| = 2k$ and $I_2 \subseteq X \cap C_2$ with $|I_2| = 2k$, and then find an $I_1 - I_2$ separator S with $\alpha(S) \leq 2k$

2-Approximation Algorithm for separators

Input: Graph G, integer k, and two sets of vertices V_1, V_2

Task: Find an (V_1, V_2)-separator S with $\alpha(S) \leq 2k$, or conclude that no (V_1, V_2)-separators with $\alpha(S) \leq k$ exist

Container with bounded α

Goal: Find a vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Container with bounded α

Goal: Find a vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

- By *iterative compression*, we can assume to have a tree decomposition TD with $\alpha(TD) = \mathcal{O}(k)$

Container with bounded α

Goal: Find a vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

- By *iterative compression*, we can assume to have a tree decomposition TD with $\alpha(TD) = \mathcal{O}(k)$

Lemma

Any vertex set S can be covered by $2\alpha(S) - 1$ bags of TD

Container with bounded α

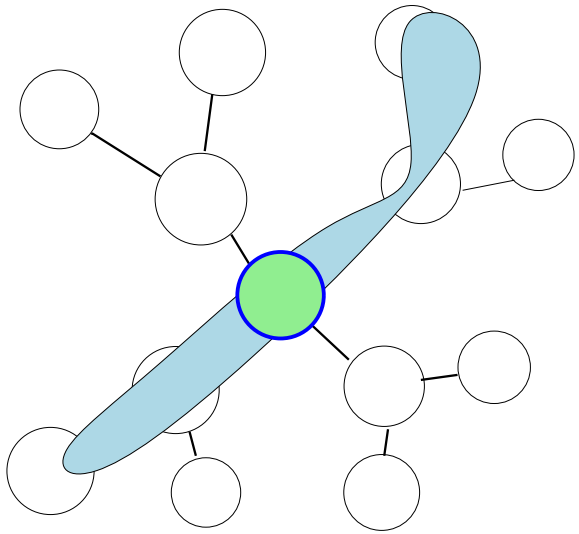
Goal: Find a vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

- By *iterative compression*, we can assume to have a tree decomposition TD with $\alpha(TD) = \mathcal{O}(k)$

Lemma

Any vertex set S can be covered by $2\alpha(S) - 1$ bags of TD

Proof: By induction on $\alpha(S)$



Container with bounded α

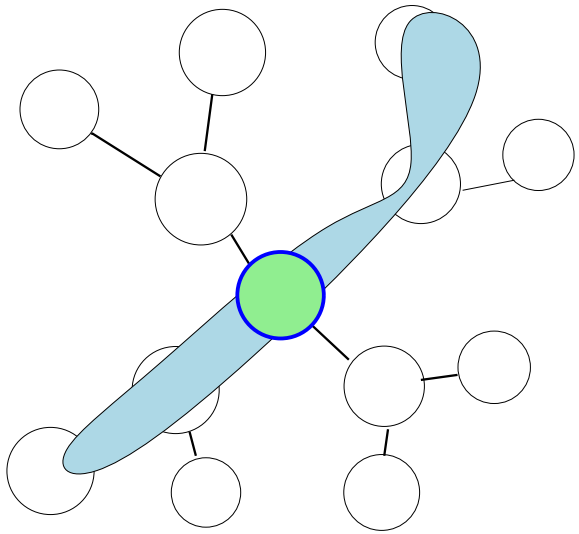
Goal: Find a vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

- By *iterative compression*, we can assume to have a tree decomposition TD with $\alpha(TD) = \mathcal{O}(k)$

Lemma

Any vertex set S can be covered by $2\alpha(S) - 1$ bags of TD

Proof: By induction on $\alpha(S)$



$\Rightarrow$ R can be guessed by guessing $\mathcal{O}(k)$ bags of TD

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Idea: Take a vertex $v \in R \setminus N(V_1 \cup V_2)$ branch on whether

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Idea: Take a vertex $v \in R \setminus N(V_1 \cup V_2)$ branch on whether

1. v goes to partial solution S_0

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Idea: Take a vertex $v \in R \setminus N(V_1 \cup V_2)$ branch on whether

1. v goes to partial solution S_0
2. v goes to V_1

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Idea: Take a vertex $v \in R \setminus N(V_1 \cup V_2)$ branch on whether

1. v goes to partial solution S_0
2. v goes to V_1
3. v goes to V_2

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Idea: Take a vertex $v \in R \setminus N(V_1 \cup V_2)$ branch on whether

1. v goes to partial solution S_0
2. v goes to V_1
3. v goes to V_2

Observation

Branches (2) and (3) decrease $\alpha(R \setminus N(V_1 \cup V_2))$.

Branching

Have: A vertex set R with $\alpha(R) \leq \mathcal{O}(k^2)$ so that $S \subseteq R$

Goal: A vertex set $R \subseteq N(V_1 \cup V_2)$ so that $S \subseteq R$

Idea: Take a vertex $v \in R \setminus N(V_1 \cup V_2)$ branch on whether

1. v goes to partial solution S_0
2. v goes to V_1
3. v goes to V_2

Observation

Branches (2) and (3) decrease $\alpha(R \setminus N(V_1 \cup V_2))$.

$\Rightarrow$ Branching tree of size $n^{2\alpha(R)}$

Linear Programming

Input: Graph G, integer k, three disjoint sets of vertices V_1, V_2, R with $R = N(V_1 \cup V_2)$

Task: Find an (V_1, V_2)-separator $S \subseteq R$ with $\alpha(S) \leq 2k$, or conclude that no such separators with $\alpha(S) \leq k$ exist

Linear Programming

Input: Graph G, integer k, three disjoint sets of vertices V_1, V_2, R with $R = N(V_1 \cup V_2)$

Task: Find an (V_1, V_2)-separator $S \subseteq R$ with $\alpha(S) \leq 2k$, or conclude that no such separators with $\alpha(S) \leq k$ exist

Variables: x_v for all vertices $v \in R$

Linear Programming

Input: Graph G, integer k, three disjoint sets of vertices V_1, V_2, R with $R = N(V_1 \cup V_2)$

Task: Find an (V_1, V_2)-separator $S \subseteq R$ with $\alpha(S) \leq 2k$, or conclude that no such separators with $\alpha(S) \leq k$ exist

Variables: x_v for all vertices $v \in R$

Separator inequalities:
$x_v + x_u \geq 1$ for all $v \in N(V_1)$, $u \in N(V_2)$ with $v - u$ path with internal vertices in $G \setminus R$

Linear Programming

Input: Graph G, integer k, three disjoint sets of vertices V_1, V_2, R with $R = N(V_1 \cup V_2)$

Task: Find an (V_1, V_2)-separator $S \subseteq R$ with $\alpha(S) \leq 2k$, or conclude that no such separators with $\alpha(S) \leq k$ exist

Variables: x_v for all vertices $v \in R$

Separator inequalities:
$x_v + x_u \geq 1$ for all $v \in N(V_1)$, $u \in N(V_2)$ with $v - u$ path with internal vertices in $G \setminus R$

Independence number inequalities:
$\sum_{v \in I} x_v \leq k$ for all independent sets $I \subseteq R$ with $|I| = 2k + 1$

Linear Programming

Input: Graph G, integer k, three disjoint sets of vertices V_1, V_2, R with $R = N(V_1 \cup V_2)$

Task: Find an (V_1, V_2)-separator $S \subseteq R$ with $\alpha(S) \leq 2k$, or conclude that no such separators with $\alpha(S) \leq k$ exist

Variables: x_v for all vertices $v \in R$

Separator inequalities:
$x_v + x_u \geq 1$ for all $v \in N(V_1)$, $u \in N(V_2)$ with $v - u$ path with internal vertices in $G \setminus R$

Independence number inequalities:
$\sum_{v \in I} x_v \leq k$ for all independent sets $I \subseteq R$ with $|I| = 2k + 1$

Lemma

Rounding a fractional solution gives a solution with independence number at most $2k$

Conclusion

- First XP approximation algorithm for tree-independence number

Conclusion

- First XP approximation algorithm for tree-independence number
- Testing tree-$\alpha(G) \leq k$ is NP-hard for every $k \geq 4$

Conclusion

- First XP approximation algorithm for tree-independence number
- Testing tree-$\alpha(G) \leq k$ is NP-hard for every $k \geq 4$
- Open problem: Complexity of testing tree-$\alpha(G) \leq k$ for $k = 2, 3$?

Thank you!