Computing Tree Decompositions with Small Independence Number

Clément Dallard[1], Fedor V. Fomin, Petr A. Golovach, Tuukka Korhonen, Martin Milanič[2]

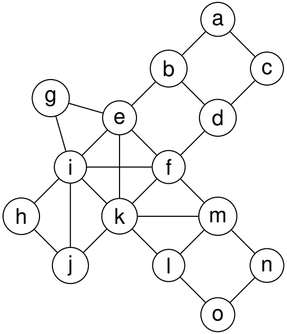
UNIVERSITY OF BERGEN

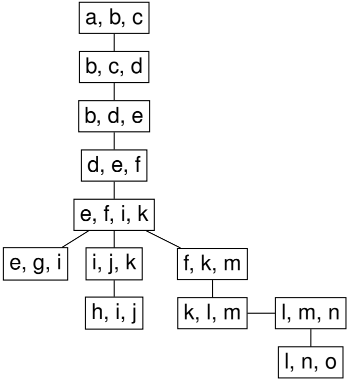
[1]University of Fribourg [2]University of Primorska

ICALP 2024

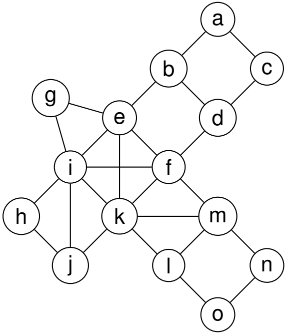
9 July 2024

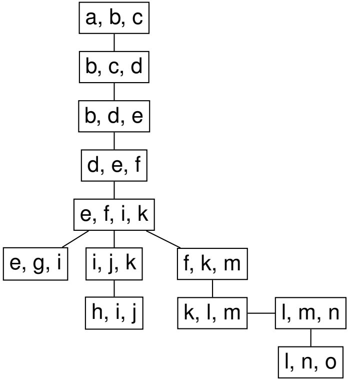
Tree Decompositions





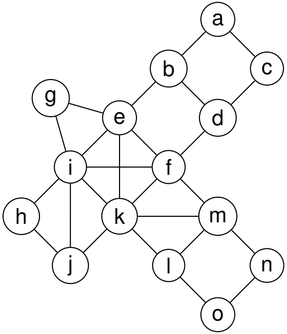
Tree Decompositions

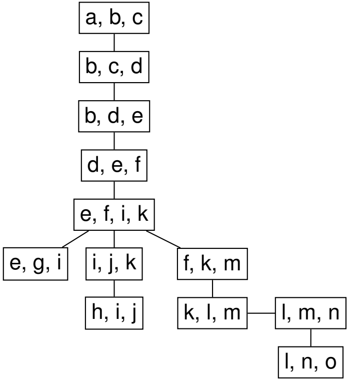




1. Every vertex should be in a bag

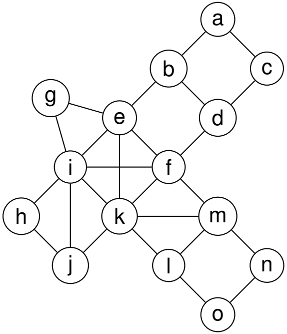
Tree Decompositions

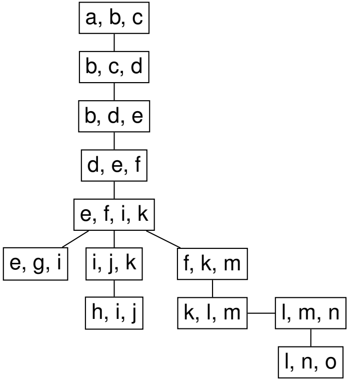




1. Every vertex should be in a bag
2. Every edge should be in a bag

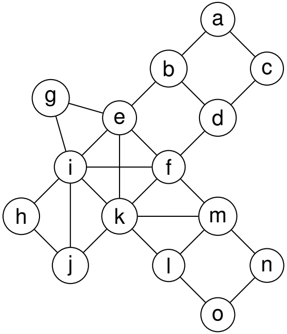
Tree Decompositions

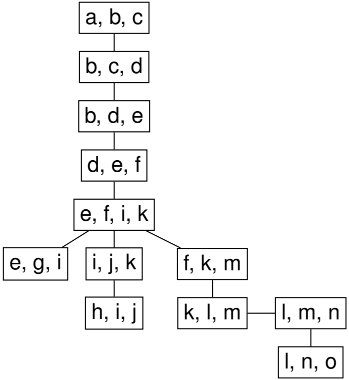




1. Every vertex should be in a bag
2. Every edge should be in a bag
3. For every vertex v, the bags containing v should form a connected subtree

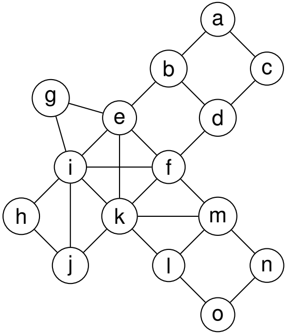
Tree Decompositions

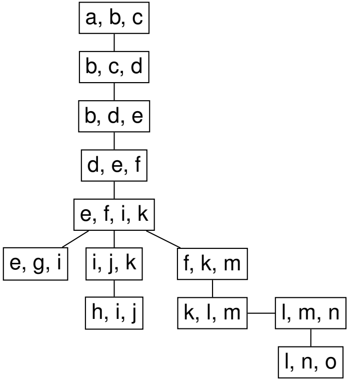




1. Every vertex should be in a bag
2. Every edge should be in a bag
3. For every vertex v, the bags containing v should form a connected subtree
4. Width = maximum bag size -1

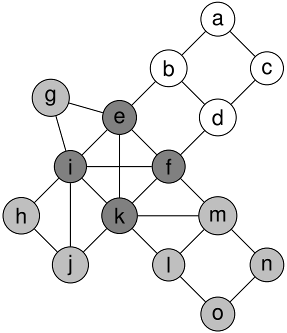
Tree Decompositions

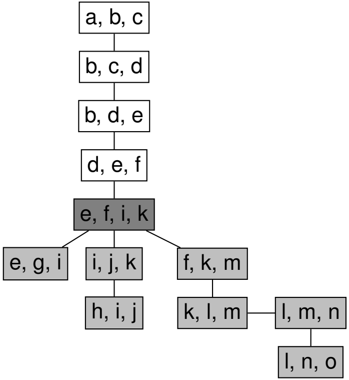




1. Every vertex should be in a bag
2. Every edge should be in a bag
3. For every vertex *v*, the bags containing *v* should form a connected subtree
4. Width = maximum bag size -1
5. Treewidth = minimum width of a tree decomposition

Dynamic programming for maximum independent set

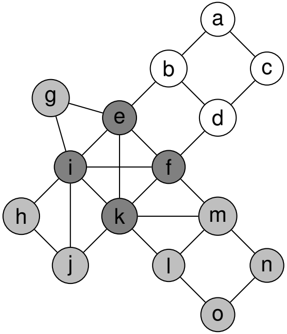


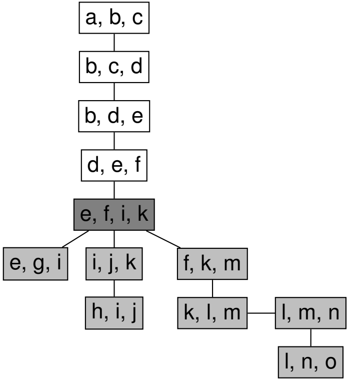


For every node t and subset $S \subseteq B_t$

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

Dynamic programming for maximum independent set



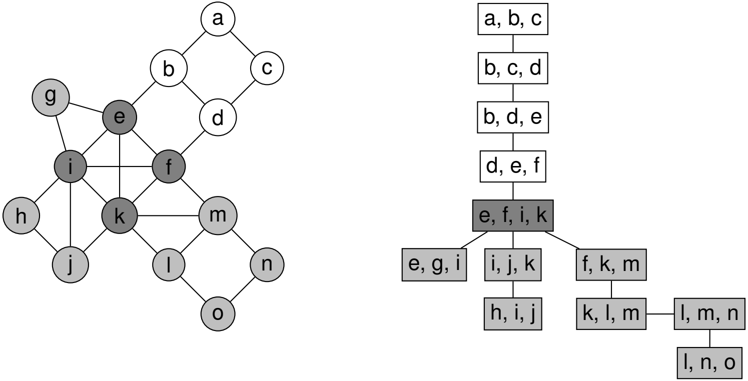


For every node t and subset $S \subseteq B_t$

$\mathrm{dp}[t][S]$ = 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 $S \subseteq B_t$

$\mathrm{dp}[t][S] =$ 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 & 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 & Golovach '20]

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 & Golovach '20]

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 & Golovach '20]

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 & Golovach '20]

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 [Yolov, SODA'18] and independently by [Dallard, Milanič, & Storgel, '21]

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 [Yolov '18]

Algorithmic Applications of Tree-independence number

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

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set [Yolov '18]
- $n^{\mathcal{O}(k)}$ time algorithms for feedback vertex set, longest induced path, and generalizations [Lima, Milanič, Muršič, Okrasa, Rzazewski, & Štorgel'24]

Algorithmic Applications of Tree-independence number

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

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set [Yolov '18]
- $n^{\mathcal{O}(k)}$ time algorithms for feedback vertex set, longest induced path, and generalizations [Lima, Milanič, Muršič, Okrasa, Rzazewski, & Štorgel'24]
- Used in Baker-like approximation schemes for geometric intersection graphs [Galby, Munaro, Yang '23]

Algorithmic Applications of Tree-independence number

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

- $\mathcal{O}(n^{k+2})$ time algorithm for maximum weight independent set [Yolov '18]
- $n^{\mathcal{O}(k)}$ time algorithms for feedback vertex set, longest induced path, and generalizations [Lima, Milanič, Muršič, Okrasa, Rzazewski, & Štorgel'24]
- Used in Baker-like approximation schemes for geometric intersection graphs [Galby, Munaro, Yang '23]

Important subroutine: Computing the 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.

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.

- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]

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.

- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]
- Almost matches the $n^{\mathcal{O}(k)}$ running time of dynamic programming

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.

- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]
- Almost matches the $n^{\mathcal{O}(k)}$ running time of dynamic programming
- Applies also to computing a slightly more general parameter called tree-μ [Yolov '18]

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.

- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]
- Almost matches the $n^{\mathcal{O}(k)}$ running time of dynamic programming
- Applies also to computing a slightly more general parameter called tree-μ [Yolov '18]

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.

- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]
- Almost matches the $n^{\mathcal{O}(k)}$ running time of dynamic programming
- Applies also to computing a slightly more general parameter called tree-μ [Yolov '18]

Hardness results:

- Assuming Gap-ETH, no $f(k) \cdot n^{o(k)}$ 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.

- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]
- Almost matches the $n^{\mathcal{O}(k)}$ running time of dynamic programming
- Applies also to computing a slightly more general parameter called tree-μ [Yolov '18]

Hardness results:

- Assuming Gap-ETH, no $f(k) \cdot n^{o(k)}$ 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.

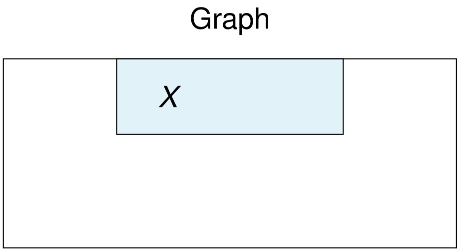
- Improves over $n^{\mathcal{O}(k^3)}$ time $\mathcal{O}(k^2)$-approximation by [Yolov '18]
- Almost matches the $n^{\mathcal{O}(k)}$ running time of dynamic programming
- Applies also to computing a slightly more general parameter called tree-μ [Yolov '18]

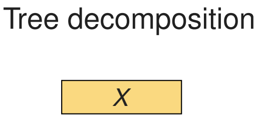
Hardness results:

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

The Algorithm

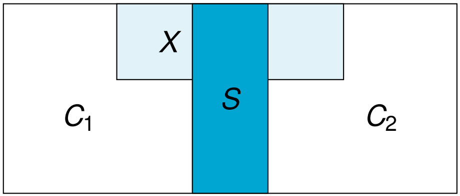
Recursive top-down construction in Robertson-Seymour fashion



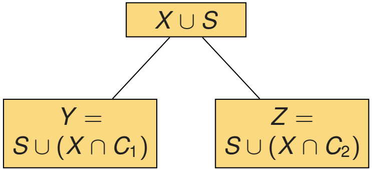


Recursive top-down construction in Robertson-Seymour fashion

Graph

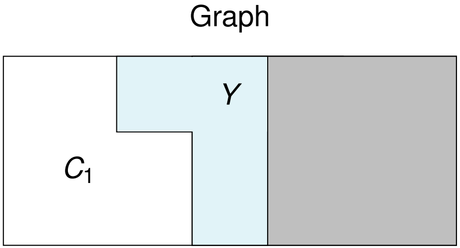


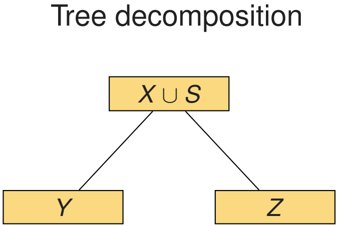
Tree decomposition



Balanced separator S with components C_1 and C_2

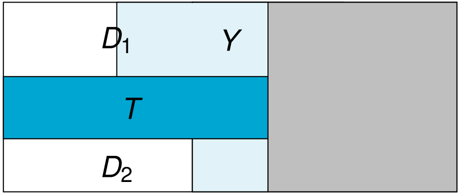
Recursive top-down construction in Robertson-Seymour fashion





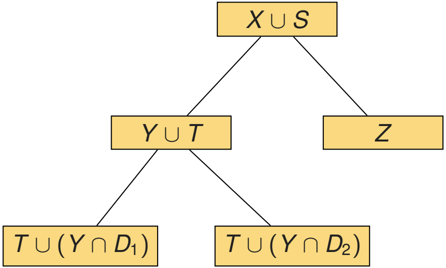
Recursive top-down construction in Robertson-Seymour fashion

Graph



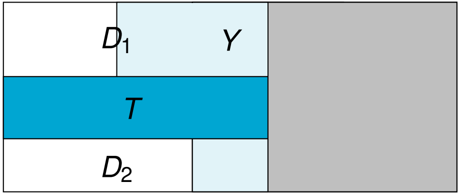
Balanced separator T with components D_1 and D_2

Tree decomposition

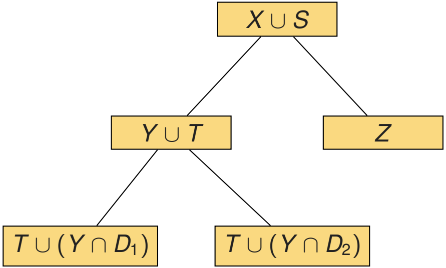


Recursive top-down construction in Robertson-Seymour fashion

Graph



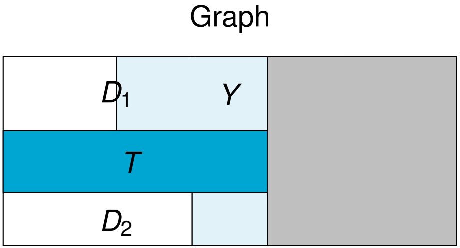
Tree decomposition



Balanced separator T with components D_1 and D_2

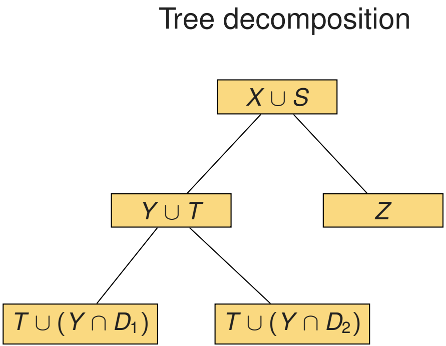
Continue recursively...

Recursive top-down construction in Robertson-Seymour fashion



Balanced separator T with components D_1 and D_2

Continue recursively...



Theorem (Informal)

If for any vertex set X with $\alpha(X) = 9k$ we can find a separation (C_1, S, C_2) so that $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, and $\alpha(X \cap C_2) \leq 7k$, then we get 11-approximation

Balanced separators

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

Task: Either (1) find a separation (C_1, S, C_2) s.t. $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, $\alpha(X \cap C_2) \leq 7k$, or (2) conclude that tree-$\alpha(G) > k$

Balanced separators

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

Task: Either (1) find a separation (C_1, S, C_2) s.t. $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, $\alpha(X \cap C_2) \leq 7k$, or (2) conclude that tree-$\alpha(G) > k$

1. Balanced separators exist because of a walking argument on a tree decomposition

Balanced separators

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

Task: Either (1) find a separation (C_1, S, C_2) s.t. $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, $\alpha(X \cap C_2) \leq 7k$, or (2) conclude that tree-$\alpha(G) > k$

1. Balanced separators exist because of a walking argument on a tree decomposition
2. Reduction from balanced separators to separators by guessing independent sets $I_i \subseteq X \cap C_i$ with $|I_i| = 2k$ and then finding $I_1 - I_2$ separator

Balanced separators

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

Task: Either (1) find a separation (C_1, S, C_2) s.t. $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, $\alpha(X \cap C_2) \leq 7k$, or (2) conclude that tree-$\alpha(G) > k$

1. Balanced separators exist because of a walking argument on a tree decomposition
2. Reduction from balanced separators to separators by guessing independent sets $I_i \subseteq X \cap C_i$ with $|I_i| = 2k$ and then finding $I_1 - I_2$ separator
3. 2-Approximation algorithm for separators

Balanced separators

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

Task: Either (1) find a separation (C_1, S, C_2) s.t. $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, $\alpha(X \cap C_2) \leq 7k$, or (2) conclude that tree-$\alpha(G) > k$

1. Balanced separators exist because of a walking argument on a tree decomposition
2. Reduction from balanced separators to separators by guessing independent sets $I_i \subseteq X \cap C_i$ with $|I_i| = 2k$ and then finding $I_1 - I_2$ separator
3. 2-Approximation algorithm for separators
   3.1 Iterative compression to guess a container with bounded α

Balanced separators

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

Task: Either (1) find a separation (C_1, S, C_2) s.t. $\alpha(S) \leq 2k$, $\alpha(X \cap C_1) \leq 7k$, $\alpha(X \cap C_2) \leq 7k$, or (2) conclude that tree-$\alpha(G) > k$

1. Balanced separators exist because of a walking argument on a tree decomposition
2. Reduction from balanced separators to separators by guessing independent sets $I_i \subseteq X \cap C_i$ with $|I_i| = 2k$ and then finding $I_1 - I_2$ separator
3. 2-Approximation algorithm for separators
   3.1 Iterative compression to guess a container with bounded α
   3.2 Branching + linear programming to find the separator

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number
- para-NP-hardness of computing tree-independence number exactly

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number
- para-NP-hardness of computing tree-independence number exactly
- Open problems:

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number
- para-NP-hardness of computing tree-independence number exactly
- Open problems:
  - Complexity of deciding tree-$\alpha(G) \leq k$ for $k = 2, 3$?

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number
- para-NP-hardness of computing tree-independence number exactly
- Open problems:
  - Complexity of deciding tree-$\alpha(G) \leq k$ for $k = 2, 3$?
  - Is deciding tree-$\alpha(G) \leq k$ for unbounded k in NP or $\sum_2^P$-hard?

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number
- para-NP-hardness of computing tree-independence number exactly
- Open problems:
  - Complexity of deciding tree-$\alpha(G) \leq k$ for $k = 2, 3$?
  - Is deciding tree-$\alpha(G) \leq k$ for unbounded k in NP or $\sum_2^P$-hard?
  - More general tree decomposition based parameters for which independent set and related problems are XP?

Conclusion

- $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time 8-approximation algorithm for tree-independence number
- para-NP-hardness of computing tree-independence number exactly
- Open problems:
  - Complexity of deciding tree-$\alpha(G) \leq k$ for $k = 2, 3$?
  - Is deciding tree-$\alpha(G) \leq k$ for unbounded k in NP or $\sum_2^P$-hard?
  - More general tree decomposition based parameters for which independent set and related problems are XP?

Thank you!