A Single-Exponential Time 2-Approximation Algorithm for Treewidth

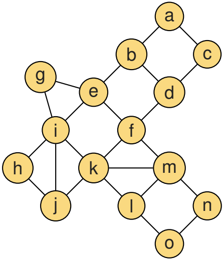
Tuukka Korhonen

University of Bergen, Norway

FOCS 2021
February 7, 2022

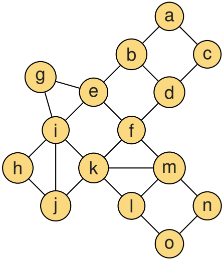
Treewidth

- Measures how close a graph is to a tree



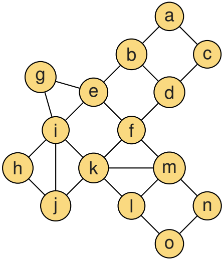
Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1



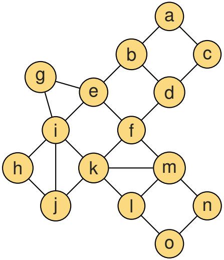
Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2



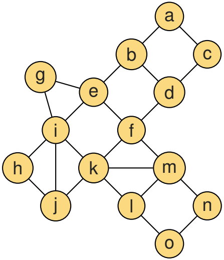
Treewidth

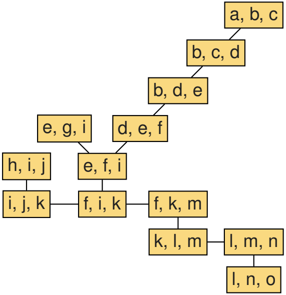
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$ -grid has treewidth n



Treewidth

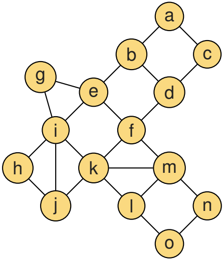
- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$ -grid has treewidth n
- Treewidth is defined as the minimum width of a *tree decomposition* [RS86, AP89, BB73, Hal76]





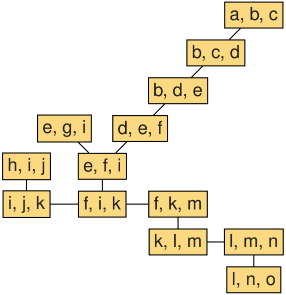
Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$ -grid has treewidth n



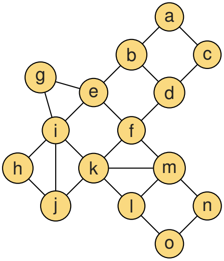
- Treewidth is defined as the minimum width of a *tree decomposition* [RS86, AP89, BB73, Hal76]

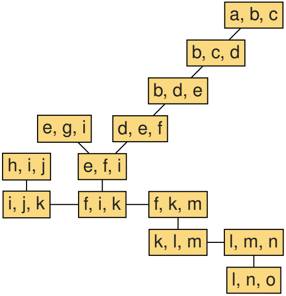
- A tree decomposition of a graph $G = (V, E)$ is a tree T of bags $B_i \subseteq V$ so that:
  1. every vertex of G is in a bag
  2. every edge of G is in a bag
  3. for each vertex of G, the bags containing it form a connected subtree of T (connectedness condition)



Treewidth

- Measures how close a graph is to a tree
  - Trees have treewidth 1
  - The example graph has treewidth 2
  - The $n \times n$ -grid has treewidth n
- Treewidth is defined as the minimum width of a *tree decomposition* [RS86, AP89, BB73, Hal76]
- A tree decomposition of a graph $G = (V, E)$ is a tree T of bags $B_i \subseteq V$ so that:
  1. every vertex of G is in a bag
  2. every edge of G is in a bag
  3. for each vertex of G, the bags containing it form a connected subtree of T (connectedness condition)
- The width of a tree decomposition is $\max |B_i| - 1$





Width 2

Why treewidth

Hundreds of results of form:

Given an n-vertex graph with a tree decomposition of width k, some combinatorial problem can be solved in time $f(k)n^c$

Why treewidth

Hundreds of results of form:

Given an n-vertex graph with a tree decomposition of width k, some combinatorial problem can be solved in time $f(k)n^c$

Often $f(k) = 2^{\mathcal{O}(k)}$ and $c = 1$

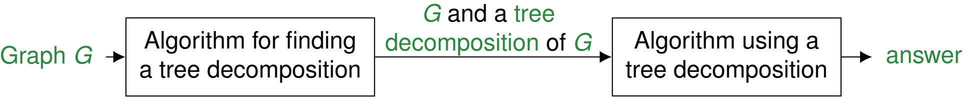
Why treewidth

Hundreds of results of form:

> Given an n-vertex graph with a tree decomposition of width k, some combinatorial problem can be solved in time $f(k)n^c$

Often $f(k) = 2^{\mathcal{O}(k)}$ and $c = 1$

Here, the algorithm will look as follows:



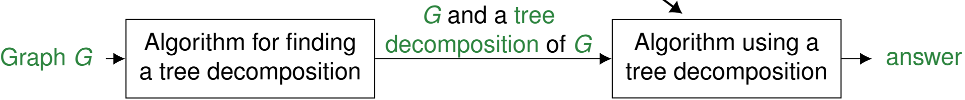
Why treewidth

Hundreds of results of form:

Given an n-vertex graph with a tree decomposition of width k, some combinatorial problem can be solved in time $f(k)n^c$

Often $f(k) = 2^{\mathcal{O}(k)}$ and $c = 1$

Here, the algorithm will look as follows:



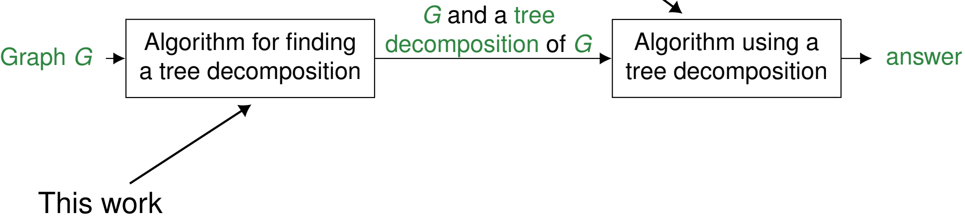
Why treewidth

Hundreds of results of form:

> Given an n-vertex graph with a tree decomposition of width k, some combinatorial problem can be solved in time $f(k)n^c$

Often $f(k) = 2^{\mathcal{O}(k)}$ and $c = 1$

Here, the algorithm will look as follows:



Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$
- Robertson and Seymour [GM XIII,'95]: 4-approximation in time $2^{\mathcal{O}(k)} n^2$

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$
- Robertson and Seymour [GM XIII,'95]: 4-approximation in time $2^{\mathcal{O}(k)} n^2$
- Bodlaender '96: Optimal in time $2^{\mathcal{O}(k^3)} n$

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$
- Robertson and Seymour [GM XIII,'95]: 4-approximation in time $2^{\mathcal{O}(k)} n^2$
- Bodlaender '96: Optimal in time $2^{\mathcal{O}(k^3)} n$
- Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk '13: 5-approximation in time $2^{\mathcal{O}(k)} n$ and 3-approximation in time $2^{\mathcal{O}(k)} n \log n$

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$
- Robertson and Seymour [GM XIII,'95]: 4-approximation in time $2^{\mathcal{O}(k)} n^2$
- Bodlaender '96: Optimal in time $2^{\mathcal{O}(k^3)} n$
- Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk '13: 5-approximation in time $2^{\mathcal{O}(k)} n$ and 3-approximation in time $2^{\mathcal{O}(k)} n \log n$
- And many others [Ree92, Lag96, FHL08, Ami10, EJT10, FTV15, FLS+18, BF21]

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$
- Robertson and Seymour [GM XIII,'95]: 4-approximation in time $2^{\mathcal{O}(k)} n^2$
- Bodlaender '96: Optimal in time $2^{\mathcal{O}(k^3)} n$
- Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk '13: 5-approximation in time $2^{\mathcal{O}(k)} n$ and 3-approximation in time $2^{\mathcal{O}(k)} n \log n$
- And many others [Ree92, Lag96, FHL08, Ami10, EJT10, FTV15, FLS+18, BF21]

Theorem (This work)

There is a $2^{\mathcal{O}(k)} n$ time 2-approximation algorithm for treewidth

Algorithms for finding tree decompositions

Long history of algorithms for finding tree decompositions

- Arnborg, Corneil, Proskurowski '87: Optimal tree decomposition in time $\mathcal{O}(n^{k+2})$
- Robertson and Seymour [GM XIII,'95]: 4-approximation in time $2^{\mathcal{O}(k)} n^2$
- Bodlaender '96: Optimal in time $2^{\mathcal{O}(k^3)} n$
- Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk '13:
  5-approximation in time $2^{\mathcal{O}(k)} n$ and 3-approximation in time $2^{\mathcal{O}(k)} n \log n$
- And many others [Ree92, Lag96, FHL08, Ami10, EJT10, FTV15, FLS$^+$18, BF21]

Theorem (This work)

There is a $2^{\mathcal{O}(k)} n$ time 2-approximation algorithm for treewidth

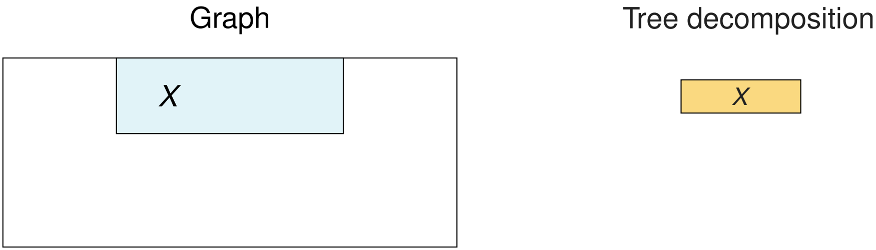
New approach for approximating treewidth

Previous algorithms

Previous approximation algorithms for treewidth [RS95, Lag96, Ree92, FHL08, Ami10, BDD^{+}16, FLS^{+}18, BF21] build the decomposition in a top-down manner, following [RS95]:

Previous algorithms

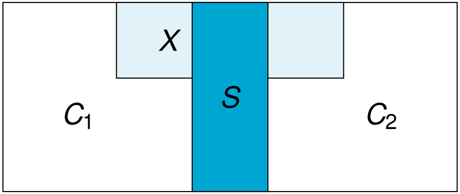
Previous approximation algorithms for treewidth [RS95, Lag96, Ree92, FHL08, Ami10, BDD+16, FLS+18, BF21] build the decomposition in a top-down manner, following [RS95]:



Previous algorithms

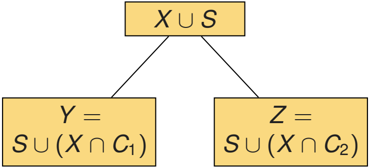
Previous approximation algorithms for treewidth [RS95, Lag96, Ree92, FHL08, Ami10, BDD$^+$16, FLS$^+$18, BF21] build the decomposition in a top-down manner, following [RS95]:

Graph



Separator S with components C_1 and C_2

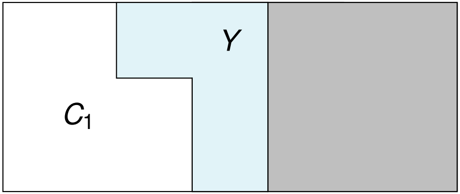
Tree decomposition



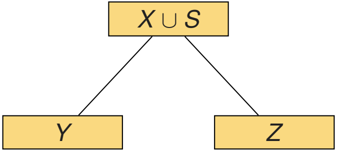
Previous algorithms

Previous approximation algorithms for treewidth [RS95, Lag96, Ree92, FHL08, Ami10, BDD$^+$16, FLS$^+$18, BF21] build the decomposition in a top-down manner, following [RS95]:

Graph



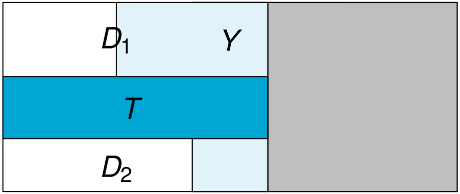
Tree decomposition



Previous algorithms

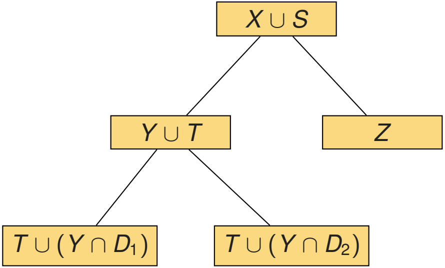
Previous approximation algorithms for treewidth
[RS95, Lag96, Ree92, FHL08, Ami10, BDD+16, FLS+18, BF21]
build the decomposition in a top-down manner, following [RS95]:

Graph



Separator T with components D_1 and D_2

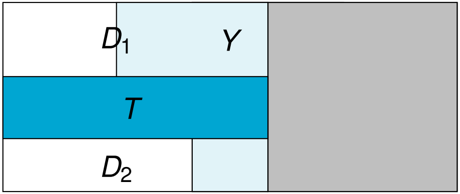
Tree decomposition



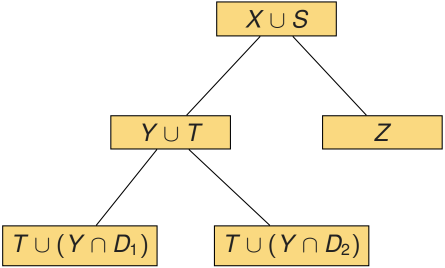
Previous algorithms

Previous approximation algorithms for treewidth [RS95, Lag96, Ree92, FHL08, Ami10, BDD+16, FLS+18, BF21] build the decomposition in a top-down manner, following [RS95]:

Graph



Tree decomposition



- Barrier at approximation ratio 3

The algorithm of this work

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w

Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$

Time complexity: $2^{\mathcal{O}(w)} n$

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w

Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$

Time complexity: $2^{\mathcal{O}(w)}n$

Let T be a tree decomposition of width w

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w

Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2\text{tw}(G) + 1$ then T can be improved by a certain improvement operation

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w

Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2\text{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w

Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$

Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2\text{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width
   - Inspired by a proof of Bellenbaum and Diestel on lean tree decompositions [BD02]

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w
Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$
Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2\text{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width
   - Inspired by a proof of Bellenbaum and Diestel on lean tree decompositions [BD02]
2. To improve the width by one, $\Omega(n)$ improvement operations may be needed

Outline

By the self-reduction technique of [Bod96] we can focus on the following:

Input: An n-vertex graph G and a tree decomposition of G of width w

Output: A tree decomposition of G of width $< w$ or conclusion that $w \leq 2\text{tw}(G) + 1$

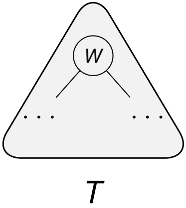
Time complexity: $2^{\mathcal{O}(w)} n$

Let T be a tree decomposition of width w

1. If $w > 2\text{tw}(G) + 1$ then T can be improved by a certain improvement operation
   - Decreases the number of bags of size $w + 1$ and does not increase the width
   - Inspired by a proof of Bellenbaum and Diestel on lean tree decompositions [BD02]
2. To improve the width by one, $\Omega(n)$ improvement operations may be needed
   - Efficient implementation by amortizatized analysis of the improvements and dynamic programming over the tree decomposition

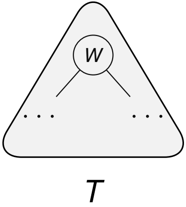
The improvement operation

- Let W be the largest bag



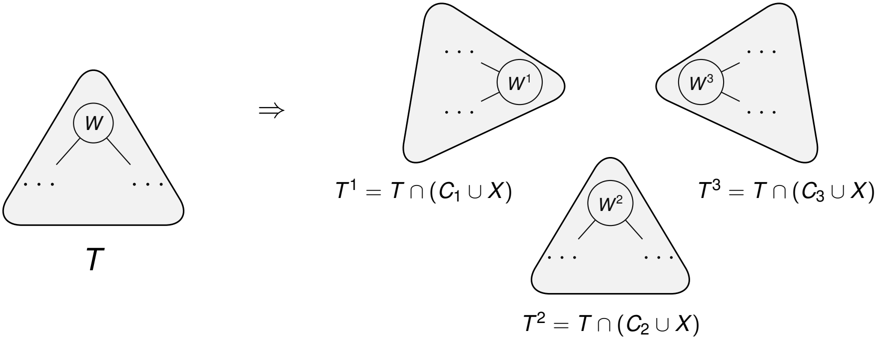
The improvement operation

- Let W be the largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of V



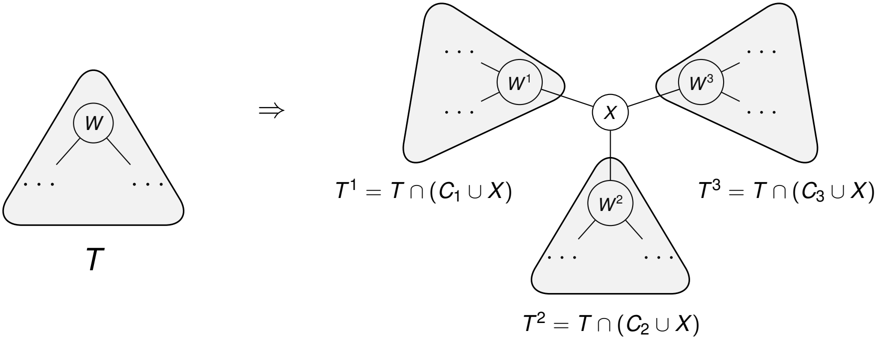
The improvement operation

- Let W be the largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of V
- For each $i \in \{1, 2, 3\}$, obtain a tree decomposition $T^i = T \cap (C_i \cup X)$ by setting $B^i = B \cap (C_i \cup X)$ for each bag B of T.



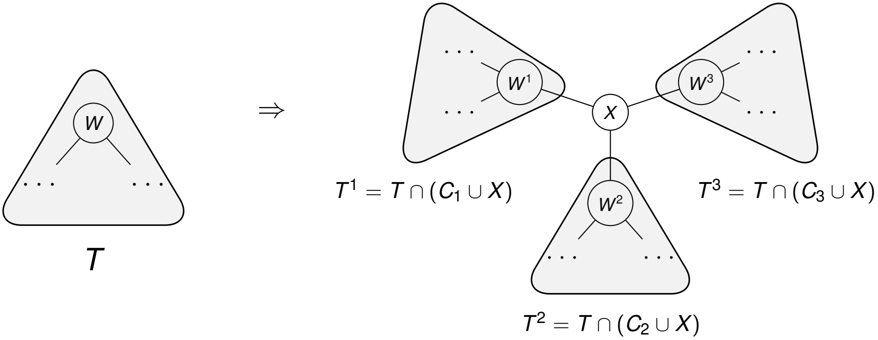
The improvement operation

- Let W be the largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of V
- For each $i \in \{1, 2, 3\}$, obtain a tree decomposition $T^i = T \cap (C_i \cup X)$ by setting $B^i = B \cap (C_i \cup X)$ for each bag B of T.
- The following is almost a tree decomposition of G:



The improvement operation

- Let W be the largest bag
- Take a small balanced separator X of W with partition (X, C_1, C_2, C_3) of V
- For each $i \in \{1, 2, 3\}$, obtain a tree decomposition $T^i = T \cap (C_i \cup X)$ by setting $B^i = B \cap (C_i \cup X)$ for each bag B of T.
- The following is almost a tree decomposition of G:



Except that vertices in X may violate the connectedness condition

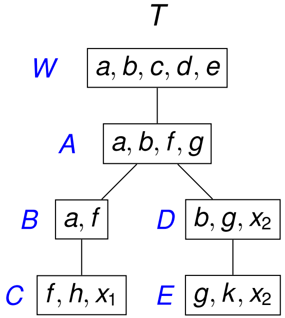
Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of X to bags

Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of X to bags

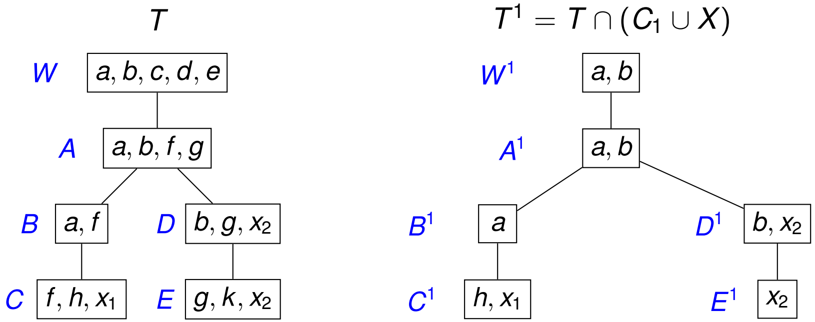
Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:



Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of X to bags

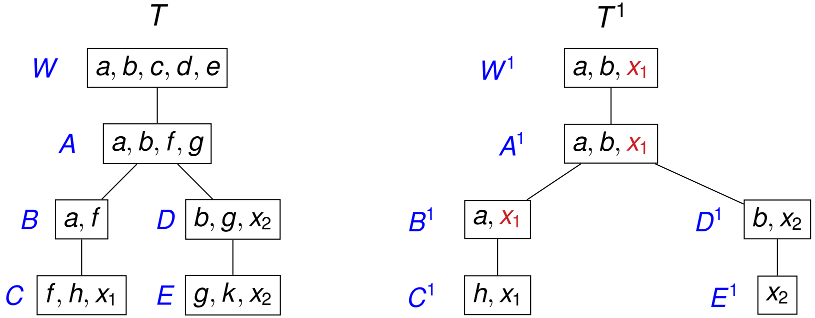
Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:



Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of X to bags

Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:

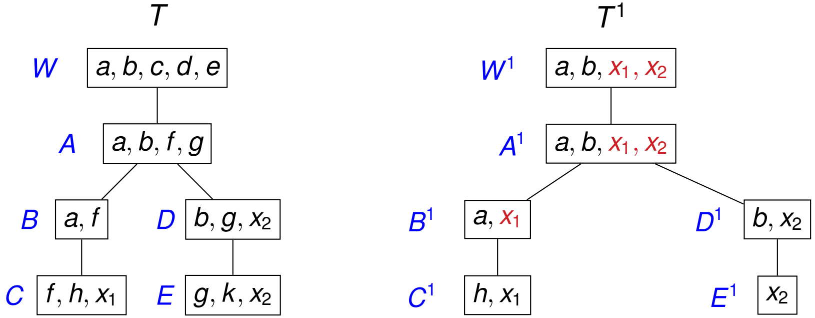


- Insert x_1 to B^1, A^1, and W^1

Fixing a tree decomposition

- Fix the connectedness condition by inserting vertices of X to bags

Example: Let $(X, C_1, C_2, C_3) = (\{x_1, x_2\}, \{a, b, h\}, \{c, d, f\}, \{e, g, k\})$ be the partition:



- Insert x_1 to B^1, A^1, and W^1
- Insert x_2 to A^1 and W^1

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

- $|B^i| = |B|$ holds only in a degenerate case where we can throw B^j for $j \neq i$ away

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

- $|B^i| = |B|$ holds only in a degenerate case where we can throw B^j for $j \neq i$ away
- For the bag W, $|W^i| < |W|$ is ensured by the definition of the balanced separator

Analysis of the improvement

- Each bag B is replaced by bags B^1, B^2, B^3

Lemma

If the balanced separator X is chosen according to specific criteria, then $|B^i| \leq |B|$ for all bags B and each i.

- $|B^i| = |B|$ holds only in a degenerate case where we can throw B^j for $j \neq i$ away
- For the bag W, $|W^i| < |W|$ is ensured by the definition of the balanced separator

$\Rightarrow$ The number of bags of size $|W|$ decreases

Conclusion

- There is a $2^{\mathcal{O}(k)}n$ time 2-approximation algorithm for treewidth

Conclusion

- There is a $2^{\mathcal{O}(k)}n$ time 2-approximation algorithm for treewidth
- Subsequently, we extended the approach to also branchwidth of symmetric submodular functions [Fomin and K., STOC'22]

Conclusion

- There is a $2^{\mathcal{O}(k)} n$ time 2-approximation algorithm for treewidth
- Subsequently, we extended the approach to also branchwidth of symmetric submodular functions [Fomin and K., STOC'22]
- Open problem: Is there a $2^{\mathcal{O}(k^c)} n^{\mathcal{O}(1)}$ time exact algorithm for treewidth, where $c < 3$? (or even a better than 2-approximation?)

The end

Thank you for your attention!

Bibliography

Eyal Amir.
Approximation algorithms for treewidth.
Algorithmica, 56(4):448–479, 2010.

Stefan Arnborg and Andrzej Proskurowski.
Linear time algorithms for NP-hard problems restricted to partial k-trees.
Discret. Appl. Math., 23(1):11–24, 1989.

Umberto Bertelè and Francesco Brioschi.
On non-serial dynamic programming.
J. Comb. Theory, Ser. A, 14(2):137–148, 1973.

Patrick Bellenbaum and Reinhard Diestel.
Two short proofs concerning tree-decompositions.
Comb. Probab. Comput., 11(6):541–547, 2002.

Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk.
A c^k n 5-approximation algorithm for treewidth.
SIAM J. Comput., 45(2):317–378, 2016.

Mahdi Belbasi and Martin Fürer.
An improvement of reed's treewidth approximation.
In *International Workshop on Algorithms and Computation*, pages 166–181. Springer, 2021.

Hans L. Bodlaender.
A linear-time algorithm for finding tree-decompositions of small treewidth.
SIAM J. Comput., 25(6):1305–1317, 1996.

Michael Elberfeld, Andreas Jakoby, and Till Tantau.
Logspace versions of the theorems of bodlaender and courcelle.
In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 143–152. IEEE Computer Society, 2010.

Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee.
Improved approximation algorithms for minimum weight vertex separators.
SIAM J. Comput., 38(2):629–657, 2008.