Efficient Data Structures And Graph Width Parameters

Marek Sokołowski

25 February 2025

Featured works

T. Korhonen, K. Majewski, W. Nadara, Mi. Pilipczuk, M. Sokołowski [FOCS '23]
Dynamic Treewidth

T. Korhonen, M. Sokołowski [STOC '24]
Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

T. Korhonen, W. Nadara, Mi. Pilipczuk, M. Sokołowski [SODA '24]
Fully Dynamic Approximation Schemes on Planar and Apex-Minor-Free Graphs

Mi. Pilipczuk, M. Sokołowski, A. Zych-Pawlewicz [STACS '22]
Compact Representation For Matrices of Bounded Twin-Width

Mi. Pilipczuk, M. Sokołowski [J. Comb. Theory B '24]
Graphs of Bounded Twin-Width Are Quasi-Polynomially χ-Bounded

Featured works

T. Korhonen, K. Majewski, W. Nadara, Mi. Pilipczuk, M. Sokołowski [FOCS '23]
Dynamic Treewidth

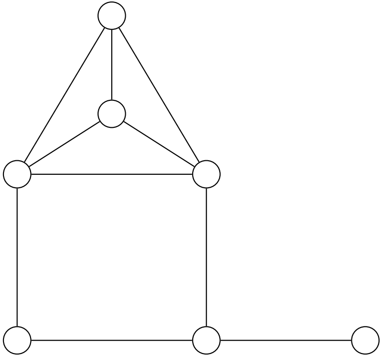
T. Korhonen, M. Sokołowski [STOC '24]
Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

T. Korhonen, W. Nadara, Mi. Pilipczuk, M. Sokołowski [SODA '24]
Fully Dynamic Approximation Schemes on Planar and Apex-Minor-Free Graphs

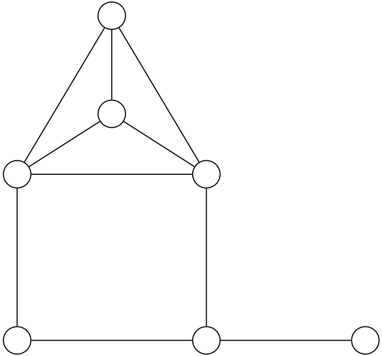
Mi. Pilipczuk, M. Sokołowski, A. Zych-Pawlewicz [STACS '22]
Compact Representation For Matrices of Bounded Twin-Width

Mi. Pilipczuk, M. Sokołowski [J. Comb. Theory B '24]
Graphs of Bounded Twin-Width Are Quasi-Polynomially χ*-Bounded*

Graphs & Graph problems

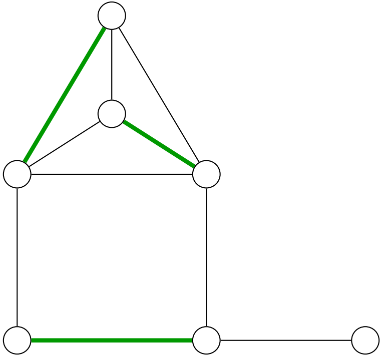


Graphs & Graph problems



n vertices, m edges

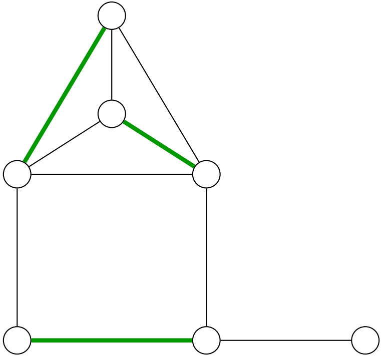
Graphs & Graph problems



n vertices, m edges

Maximum Matching

Graphs & Graph problems



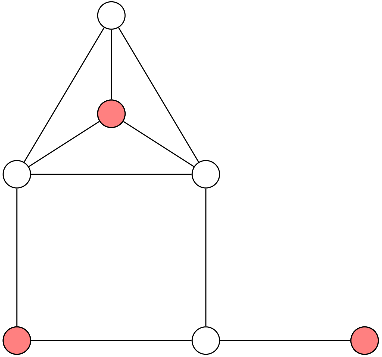
n vertices, m edges

MAXIMUM MATCHING

Easy!

[Edmonds '61]

Graphs & Graph problems



n vertices, m edges

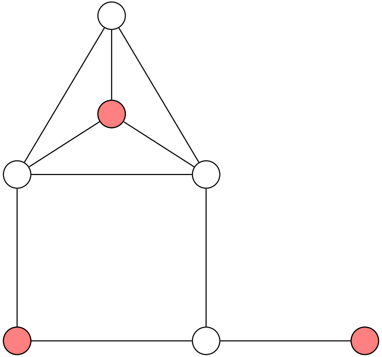
MAXIMUM MATCHING

Easy!

[Edmonds '61]

MAXIMUM INDEPENDENT SET

Graphs & Graph problems



n vertices, m edges

Maximum Matching

Easy!

[Edmonds '61]

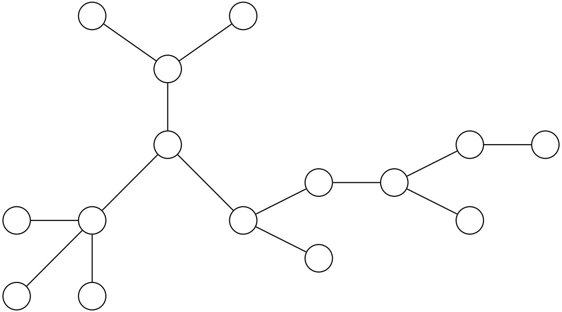
Maximum Independent Set

NP-hard!

[Cook '71, Karp '72, Levin '73]

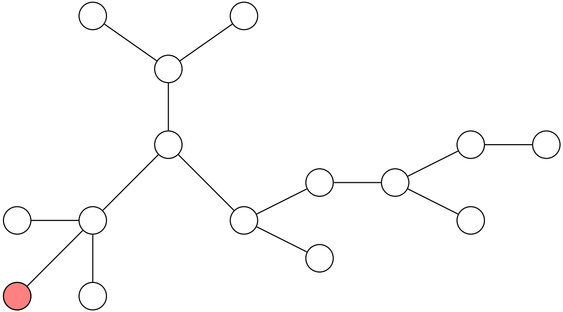
Trees

Maximum Independent Set is NP-hard in general... But becomes easy on **trees**!



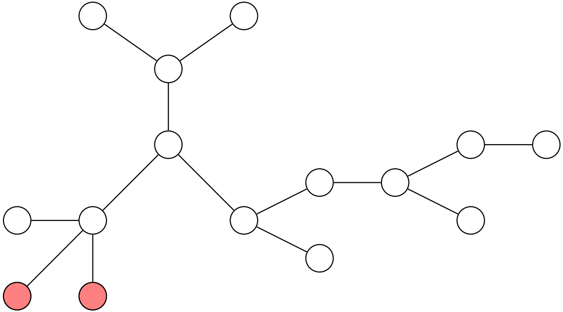
Trees

Maximum Independent Set is NP-hard in general. . . But becomes easy on **trees**!



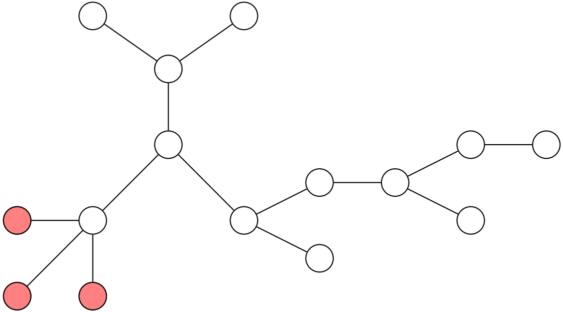
Trees

MAXIMUM INDEPENDENT SET is NP-hard in general... But becomes easy on **trees**!



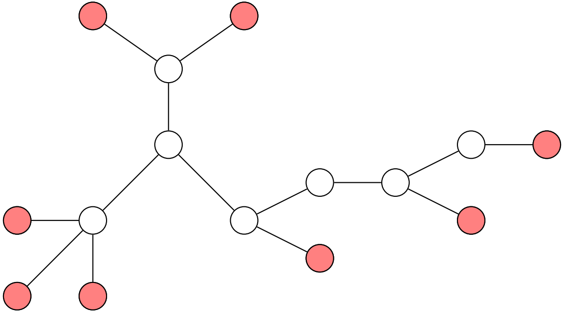
Trees

MAXIMUM INDEPENDENT SET is NP-hard in general... But becomes easy on **trees**!



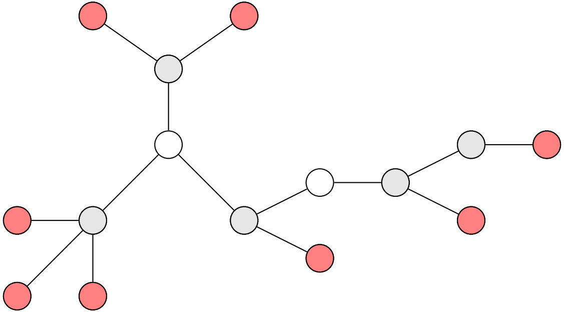
Trees

Maximum Independent Set is NP-hard in general... But becomes easy on **trees**!



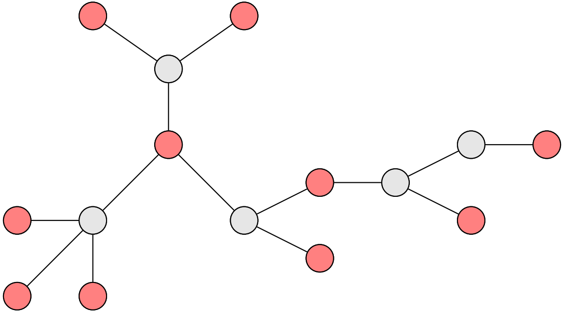
Trees

Maximum Independent Set is NP-hard in general. . . But becomes easy on **trees**!



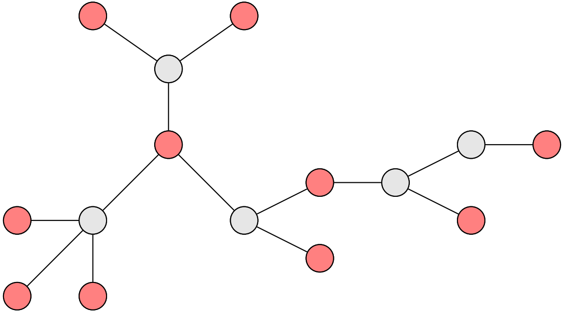
Trees

MAXIMUM INDEPENDENT SET is NP-hard in general... But becomes easy on **trees**!



Trees

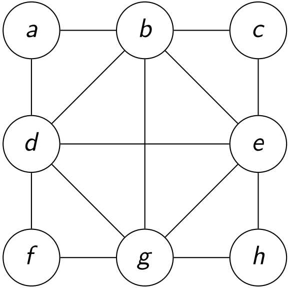
Maximum Independent Set is NP-hard in general... But becomes easy on **trees**!



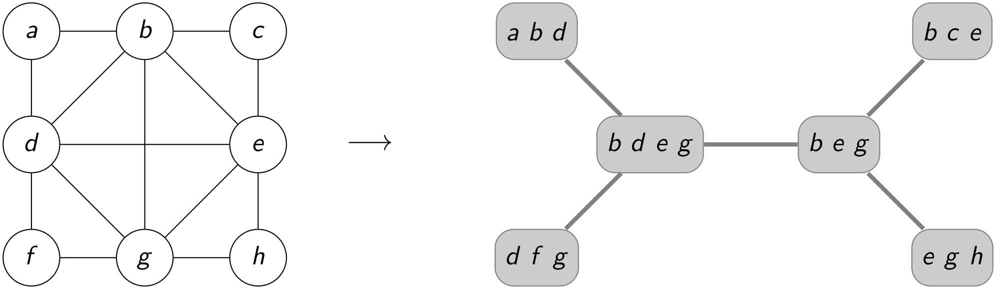
Question

Maybe some hard problems can be solved efficiently on **more general tree-like** graphs?

Treewidth

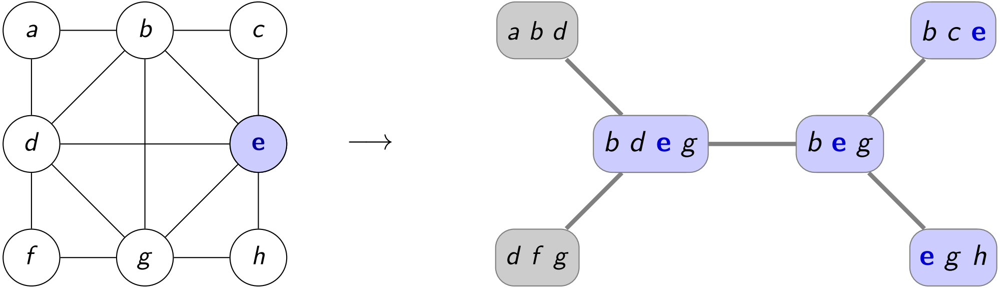


Treewidth



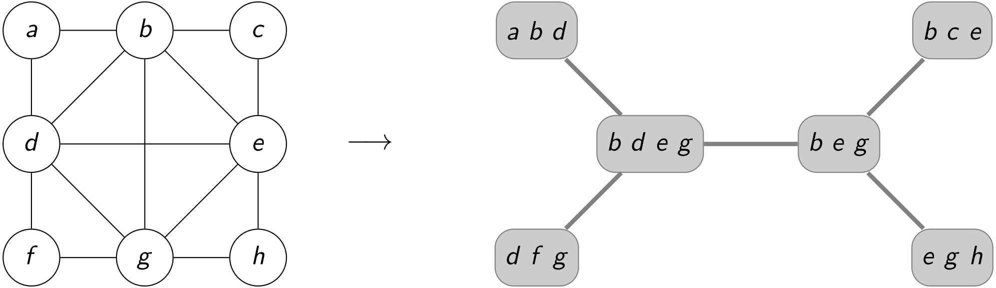
tree decomposition

Treewidth



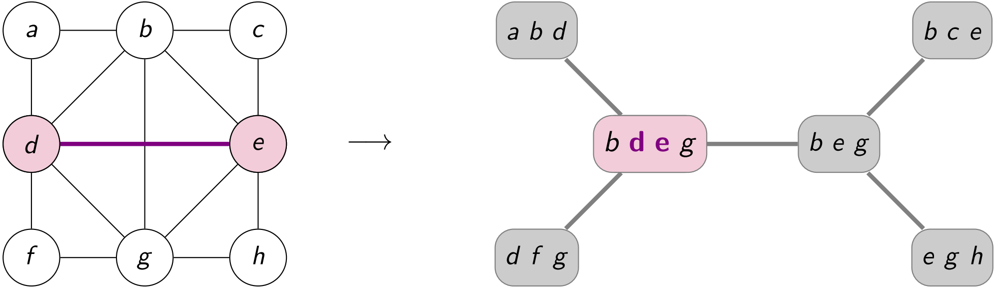
- Each **vertex** in a non-empty connected subgraph of the decomposition

Treewidth



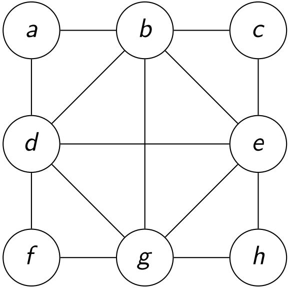
- Each **vertex** in a non-empty connected subgraph of the decomposition

Treewidth

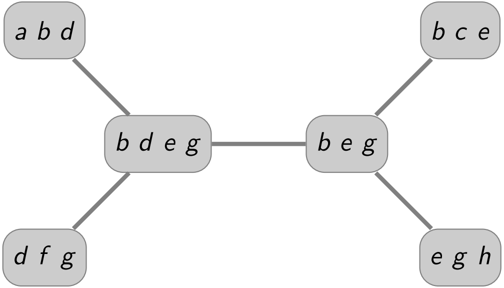


- Each **vertex** in a non-empty connected subgraph of the decomposition
- Each **edge** $uv \implies$ both u and v in some common bag of the decomposition

Treewidth

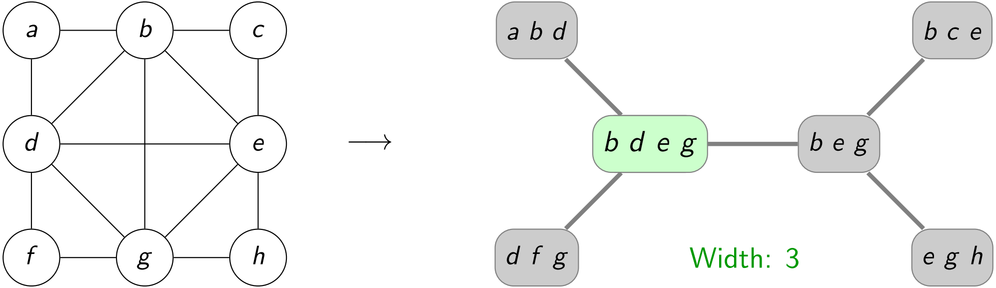


$\longrightarrow$



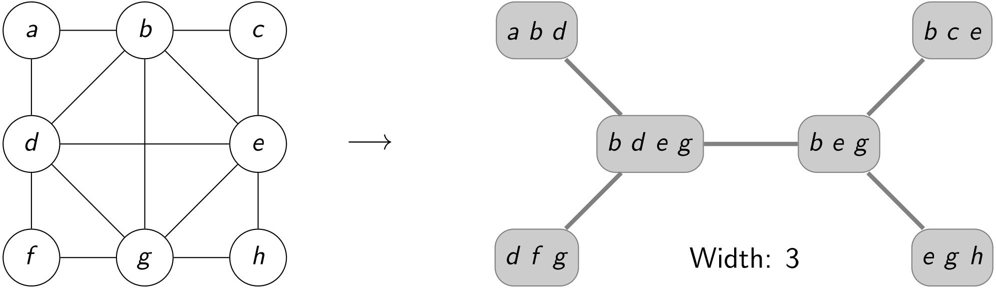
- Each **vertex** in a non-empty connected subgraph of the decomposition
- Each **edge** $uv \implies$ both u and v in some common bag of the decomposition

Treewidth



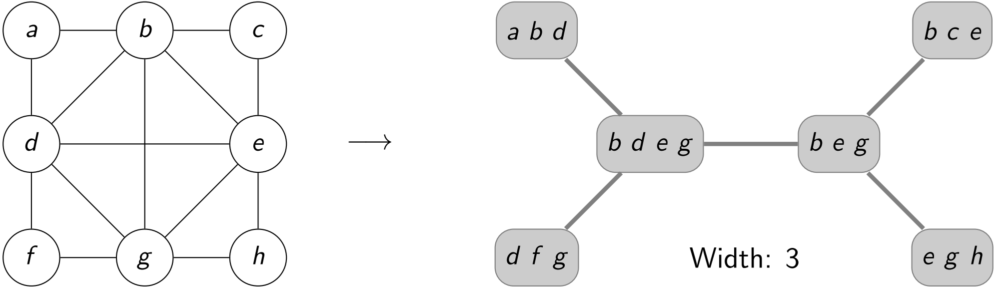
- Each **vertex** in a non-empty connected subgraph of the decomposition
- Each **edge** $uv \implies$ both u and v in some common bag of the decomposition
- **Width:** maximum bag size, minus 1

Treewidth



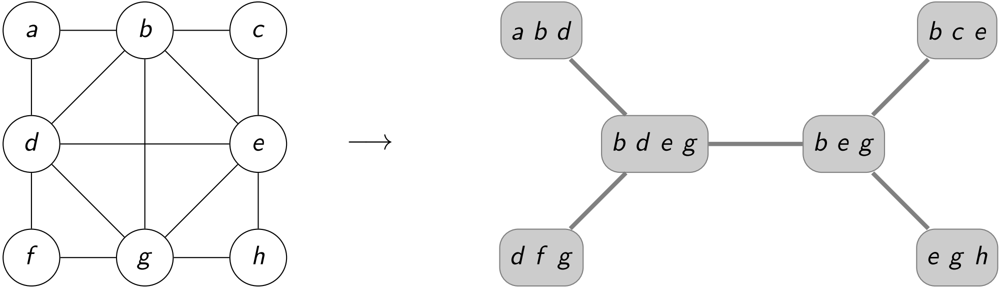
- Each **vertex** in a non-empty connected subgraph of the decomposition
- Each **edge** $uv \implies$ both u and v in some common bag of the decomposition
- **Width:** maximum bag size, minus 1

Treewidth



- Each **vertex** in a non-empty connected subgraph of the decomposition
- Each **edge** $uv \implies$ both u and v in some common bag of the decomposition
- **Width:** maximum bag size, minus 1
- **Treewidth:** minimum possible width of a tree decomposition

Treewidth



Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w

Then: Maximum Independent Set can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: Maximum Independent Set can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Problem: Usually we don't have a tree decomposition of a graph beforehand.

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: Maximum Independent Set can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Problem: Usually we don't have a tree decomposition of a graph beforehand.

Tree decomposition algorithms

Given an n-vertex graph G of treewidth w, we can **find** a tree decomposition of G...

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: MAXIMUM INDEPENDENT SET can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Problem: Usually we don't have a tree decomposition of a graph beforehand.

Tree decomposition algorithms

Given an n-vertex graph G of treewidth w, we can **find** a tree decomposition of G...

	Width guarantee	**Time**
[Robertson, Seymour '86]	$4w+3$	$2^{\mathcal{O}(w)} \cdot n^2$

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: Maximum Independent Set can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Problem: Usually we don't have a tree decomposition of a graph beforehand.

Tree decomposition algorithms

Given an n-vertex graph G of treewidth w, we can **find** a tree decomposition of G...

	Width guarantee	Time
[Robertson, Seymour '86]	$4w + 3$	$2^{\mathcal{O}(w)} \cdot n^2$
[Bodlaender '96]	w	$2^{\mathcal{O}(w^3)} \cdot n$

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: MAXIMUM INDEPENDENT SET can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Problem: Usually we don't have a tree decomposition of a graph beforehand.

Tree decomposition algorithms

Given an n-vertex graph G of treewidth w, we can **find** a tree decomposition of G...

	Width guarantee	**Time**
[Robertson, Seymour '86]	$4w + 3$	$2^{\mathcal{O}(w)} \cdot n^2$
[Bodlaender '96]	w	$2^{\mathcal{O}(w^3)} \cdot n$
[Bodlaender et al. '16]	$5w + 4$	$2^{\mathcal{O}(w)} \cdot n$

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: MAXIMUM INDEPENDENT SET can be solved in time $2^{\mathcal{O}(w)} \cdot n$

Problem: Usually we don't have a tree decomposition of a graph beforehand.

Tree decomposition algorithms

Given an n-vertex graph G of treewidth w, we can **find** a tree decomposition of G...

	Width guarantee	**Time**
[Robertson, Seymour '86]	$4w + 3$	$2^{\mathcal{O}(w)} \cdot n^2$
[Bodlaender '96]	w	$2^{\mathcal{O}(w^3)} \cdot n$
[Bodlaender et al. '16]	$5w + 4$	$2^{\mathcal{O}(w)} \cdot n$
[Korhonen '21]	$2w + 1$	$2^{\mathcal{O}(w)} \cdot n$

Treewidth

Treewidth is great!

Given: n-vertex graph G and its tree decomposition of width w
Then: Maximum Independent Set can be solved in time $2^{\mathcal{O}(w)} \cdot n$

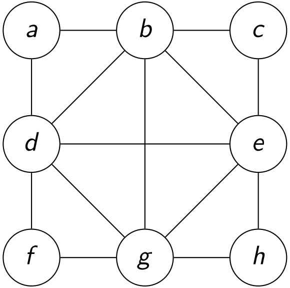
Problem: Usually we don't have a tree decomposition of a graph beforehand.

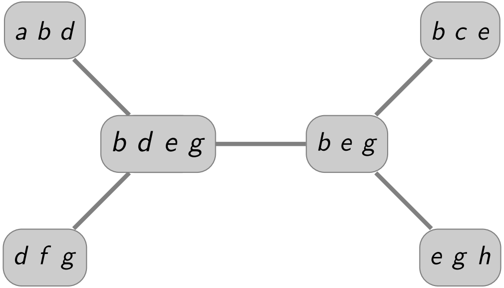
Tree decomposition algorithms

Given an n-vertex graph G of treewidth w, we can **find** a tree decomposition of G...

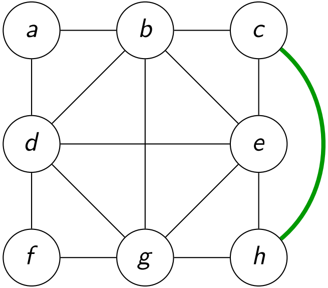
	Width guarantee	**Time**
[Robertson, Seymour '86]	$4w+3$	$2^{\mathcal{O}(w)} \cdot n^2$
[Bodlaender '96]	w	$2^{\mathcal{O}(w^3)} \cdot n$
[Bodlaender et al. '16]	$5w+4$	$2^{\mathcal{O}(w)} \cdot n$
[Korhonen '21]	$2w+1$	$2^{\mathcal{O}(w)} \cdot n$
[Korhonen, Lokshtanov '23]	w	$2^{\mathcal{O}(w^2)} \cdot n$

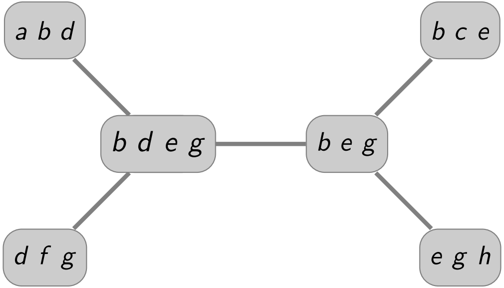
Suddenly. . .



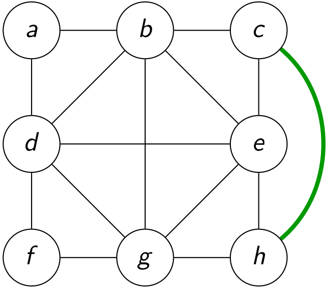


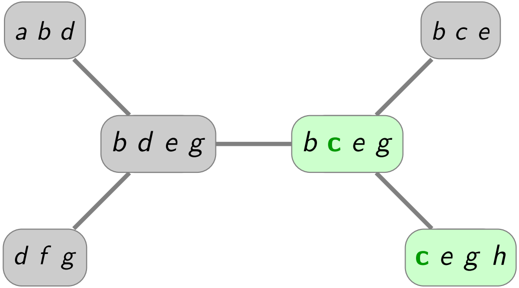
Suddenly. . .



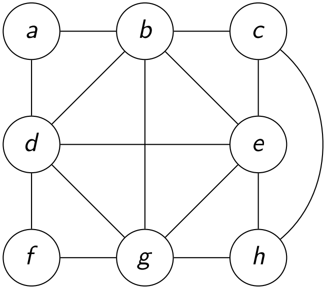


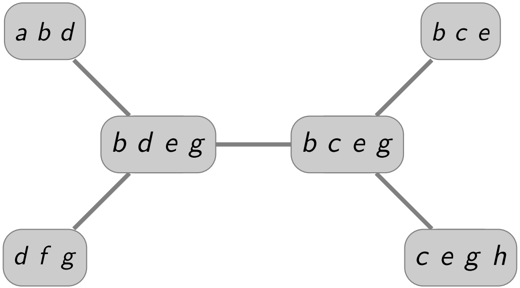
Suddenly...



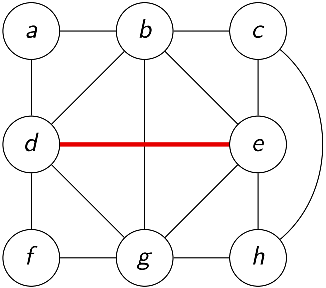


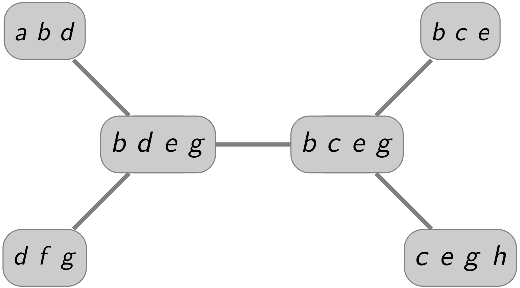
Suddenly. . .



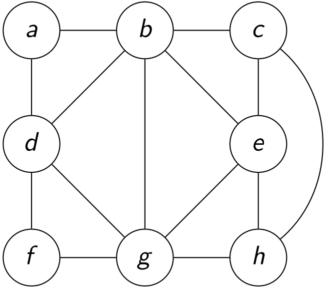


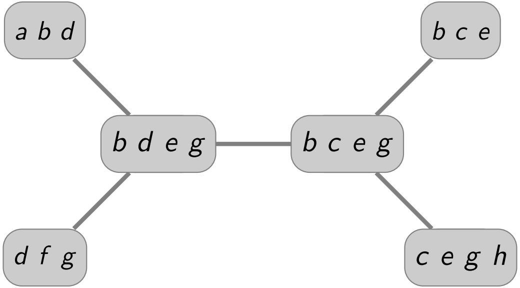
Suddenly. . .



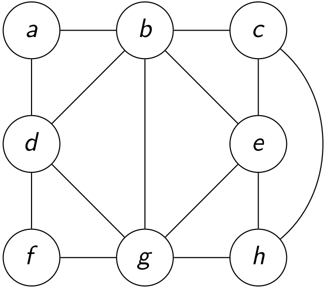


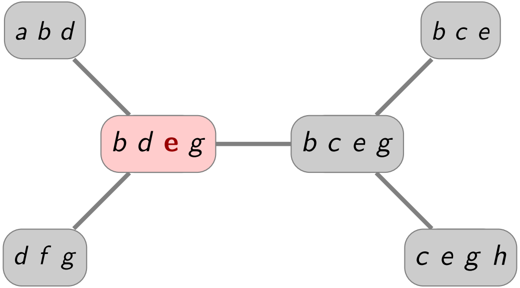
Suddenly. . .



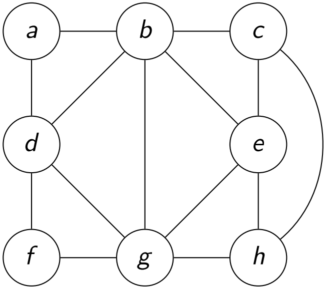


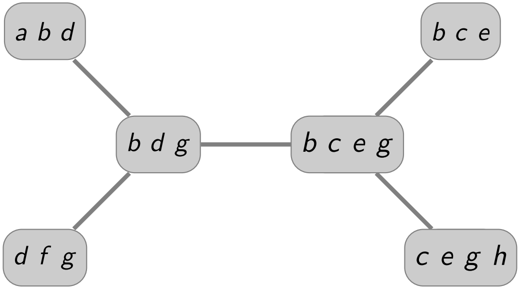
Suddenly...



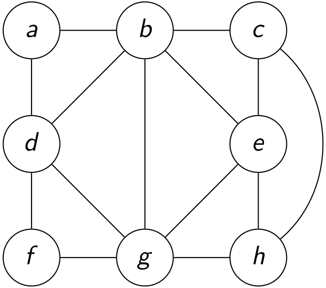


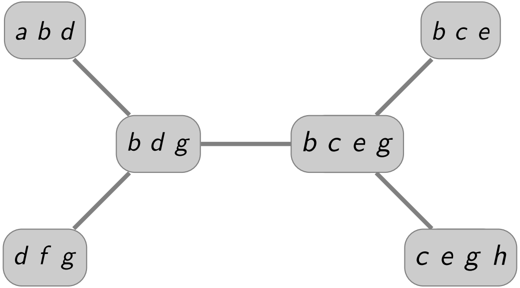
Suddenly. . .





Suddenly. . .





Problem

How to maintain tree decompositions of **dynamic graphs**?

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

In a **dynamic graph** G with n vertices of treewidth w ...

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

In a **dynamic graph** G with n vertices of treewidth w ...

We maintain: a tree decomposition of G of width at most $6w + 5$ ...

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

In a **dynamic graph** G with n vertices of treewidth w ...

We maintain: a tree decomposition of G of width at most $6w + 5$ ...

Initialization time: $2^{w^{\mathcal{O}(1)}} \cdot n$

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

In a **dynamic graph** G with n vertices of treewidth $w \ldots$

We maintain: a tree decomposition of G of width at most $6w + 5 \ldots$

Initialization time: $2^{w^{\mathcal{O}(1)}} \cdot n$

Update time: $2^{w^{\mathcal{O}(1)} \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

Dynamic Treewidth

Main result

In a **dynamic graph** G with n vertices of treewidth $w \ldots$

We maintain: a tree decomposition of G of width at most $6w + 5 \ldots$

Initialization time: $2^{w^{\mathcal{O}(1)}} \cdot n$

Update time: $2^{w^{\mathcal{O}(1)} \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

$$\log^{1000} n \quad \ll \quad \mathbf{2^{\sqrt{\log n \log \log n}}} \quad \ll \quad n^{0.001}$$

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

In a **dynamic graph** G with n vertices of treewidth w ...

We maintain: a tree decomposition of G of width at most $6w + 5$ ...

Initialization time: $2^{w^{\mathcal{O}(1)}} \cdot n$

Update time: $2^{w^{\mathcal{O}(1)} \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

Extension

We can also dynamically solve any decision/optimization problem expressible in CMSO_2 logic.

Dynamic Treewidth

Korhonen, Majewski, Nadara, Pilipczuk, **Sokołowski** [FOCS '23]

DYNAMIC TREEWIDTH

Main result

In a **dynamic graph** G with n vertices of treewidth $w\ldots$

We maintain: a tree decomposition of G of width at most $6w + 5\ldots$

Initialization time: $2^{w^{\mathcal{O}(1)}} \cdot n$

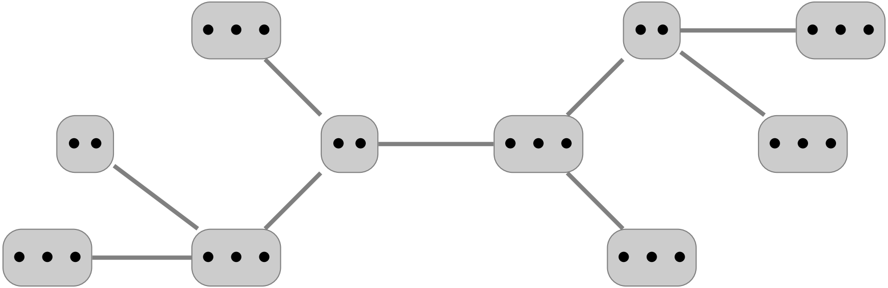
Update time: $2^{w^{\mathcal{O}(1)} \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

Extension

We can also dynamically solve any decision/optimization problem expressible in CMSO_2 logic.

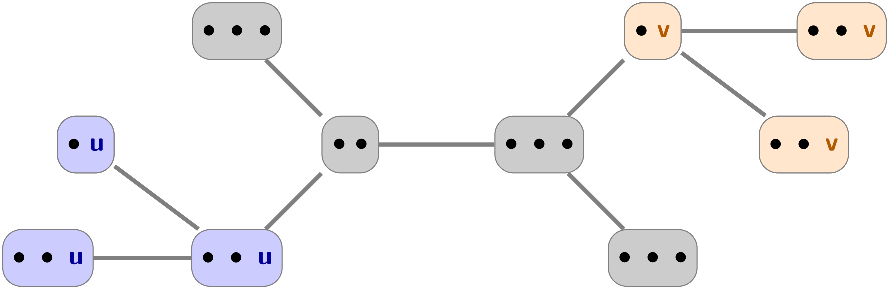
MAX MATCHING, MAX INDEPENDENT SET, LONGEST PATH, HAMILTONIAN CYCLE...

Approach



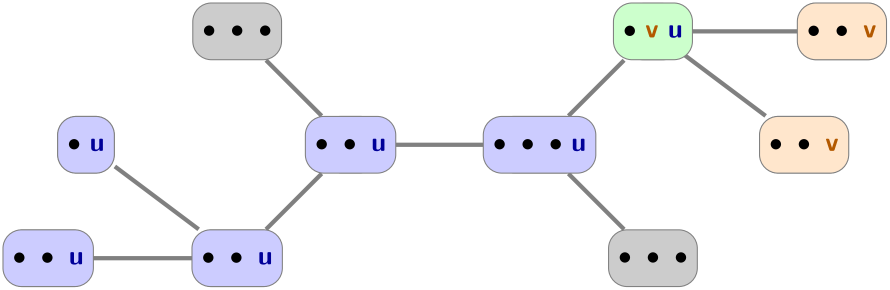
- Maintain a **low-diameter** tree decomposition of the graph ($2^{\mathcal{O}(\sqrt{\log n \log \log n})}$)

Approach



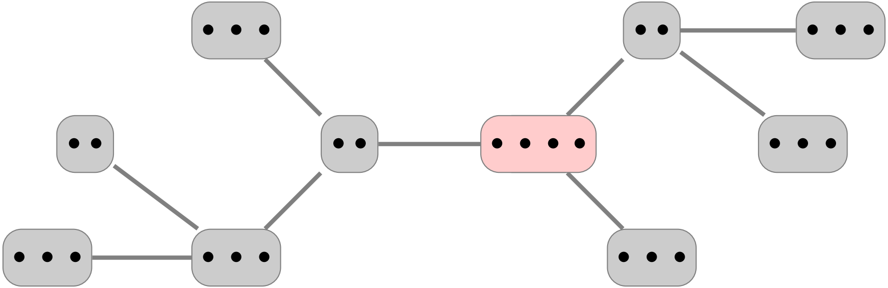
- Maintain a **low-diameter** tree decomposition of the graph ($2^{\mathcal{O}(\sqrt{\log n \log\log n})}$)
- Edge **uv** added?

Approach



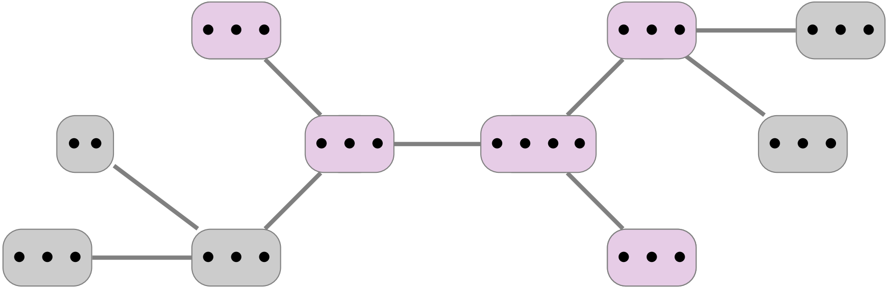
- Maintain a **low-diameter** tree decomposition of the graph $(2^{\mathcal{O}(\sqrt{\log n \log \log n})})$
- Edge **uv** added? $\implies$ Insert **u** into additional bags

Approach



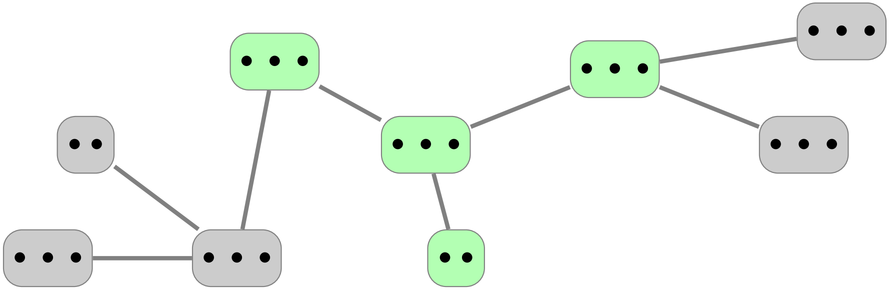
- Maintain a **low-diameter** tree decomposition of the graph $(2^{\mathcal{O}(\sqrt{\log n \log\log n})})$
- Edge **uv** added? $\implies$ Insert **u** into additional bags
- A bag **too large**?

Approach



- Maintain a **low-diameter** tree decomposition of the graph $(2^{\mathcal{O}(\sqrt{\log n \log \log n})})$
- Edge **uv** added? $\implies$ Insert **u** into additional bags
- A bag **too large**? $\implies$ Recompute a **local neighborhood** of the bag

Approach



- Maintain a **low-diameter** tree decomposition of the graph $(2^{\mathcal{O}(\sqrt{\log n \log\log n})})$
- Edge **uv** added? $\Longrightarrow$ Insert **u** into additional bags
- A bag **too large**? $\Longrightarrow$ Recompute a **local neighborhood** of the bag

Treewidth, but for denser graphs

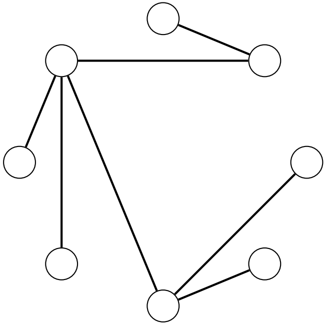
Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!

Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!

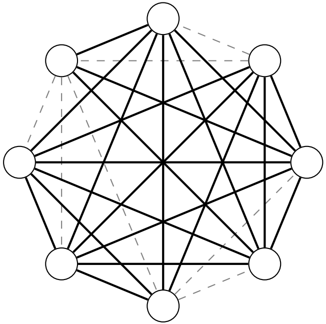


trees

Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!

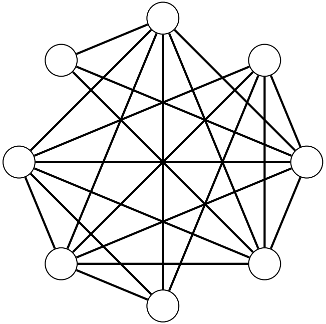


complements of trees

Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs...

But there also exist **dense** tree-like graphs!

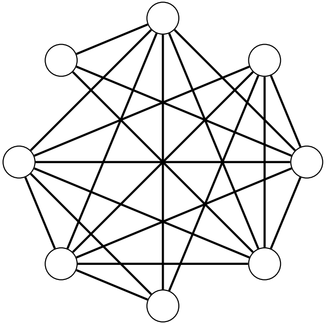


complements of trees

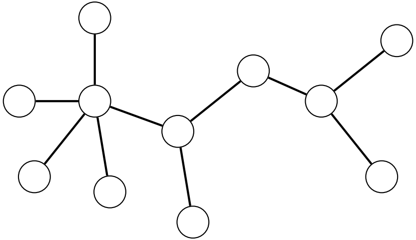
Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!



complements of trees

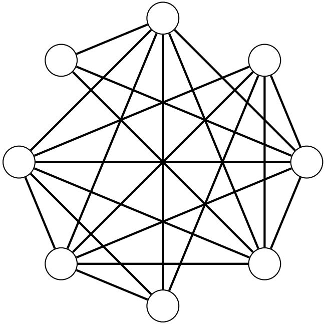


trees

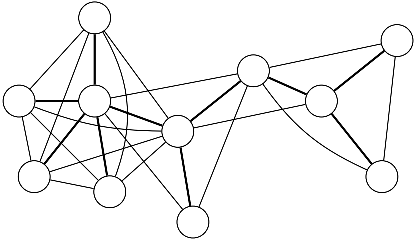
Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!



complements of trees



squares of trees

Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!

Solution

Equivalent notions of **cliquewidth** [Courcelle et al. '93] and **rankwidth** [Oum, Seymour '06].

Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs. . .

But there also exist **dense** tree-like graphs!

Solution

Equivalent notions of **cliquewidth** [Courcelle et al. '93] and **rankwidth** [Oum, Seymour '06].

Rankwidth is great!

Given: n-vertex graph G and its **rank decomposition** of width w
Then: Maximum Independent Set can be solved in time $2^{f(w)} \cdot n$

Treewidth, but for denser graphs

Issue: treewidth applicable only to **sparse** graphs...

But there also exist **dense** tree-like graphs!

Solution

Equivalent notions of **cliquewidth** [Courcelle et al. '93] and **rankwidth** [Oum, Seymour '06].

Rankwidth is great!

Given: n-vertex graph G and its **rank decomposition** of width w
Then: Maximum Independent Set can be solved in time $2^{f(w)} \cdot n$

Also Max Clique, Min Dominating Set, Longest Induced Path, ...

Rankwidth

Rankwidth is great!

Given: n-vertex graph G and its **rank decomposition** of width w
Then: Maximum Independent Set can be solved in time $2^{f(w)} \cdot n$

Same problem: Need to compute a rank decomposition.

Rankwidth

Rankwidth is great!

Given: n-vertex graph G and its **rank decomposition** of width w
Then: Maximum Independent Set can be solved in time $2^{f(w)} \cdot n$

Same problem: Need to compute a rank decomposition.

Rank decomposition algorithms

Given an n-vertex graph G of rankwidth w, we can **find** a rank decomposition of G...

	Width guarantee	**Time**
[Oum, Seymour '06]	$3w+1$	$2^{\mathcal{O}(w)} \cdot n^9$
[Oum '08]	$3w-1$	$f(w) \cdot n^3$
[Jeong, Kim, Oum '21]	w	$f(w) \cdot n^3$
[Fomin, Korhonen '22]	w	$f(w) \cdot n^2$

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

ALMOST-LINEAR TIME PARAMETERIZED ALGORITHM FOR RANKWIDTH
VIA DYNAMIC RANKWIDTH

Main result

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

ALMOST-LINEAR TIME PARAMETERIZED ALGORITHM FOR RANKWIDTH VIA DYNAMIC RANKWIDTH

Main result

In a **dynamic graph** G with n vertices and m edges of rankwidth w ...

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

ALMOST-LINEAR TIME PARAMETERIZED ALGORITHM FOR RANKWIDTH VIA DYNAMIC RANKWIDTH

Main result

In a **dynamic graph** G with n vertices and m edges of rankwidth $w \dots$

We maintain: a rank decomposition of G of width at most $4w \dots$

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

ALMOST-LINEAR TIME PARAMETERIZED ALGORITHM FOR RANKWIDTH VIA DYNAMIC RANKWIDTH

Main result

In a **dynamic graph** G with n vertices and m edges of rankwidth $w \dots$

We maintain: a rank decomposition of G of width at most $4w \dots$

Initialization time: $2^{f(w)} \cdot n \log^2 n$

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

Main result

In a **dynamic graph** G with n vertices and m edges of rankwidth w . . .

We maintain: a rank decomposition of G of width at most $4w$ . . .

Initialization time: $2^{f(w)} \cdot n \log^2 n$

Update time: $2^{f(w) \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

Main result

In a **dynamic graph** G with n vertices and m edges of rankwidth $w \ldots$

We maintain: a rank decomposition of G of width at most $4w \ldots$

Initialization time: $2^{f(w)} \cdot n \log^2 n$

Update time: $2^{f(w) \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

Extension

We can also dynamically solve any decision/optimization problem expressible in CMSO_1 logic.

Dynamic Rankwidth

Korhonen, **Sokołowski** [STOC '24]

Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

Main result

In a **dynamic graph** G with n vertices and m edges of rankwidth $w\ldots$

We maintain: a rank decomposition of G of width at most $4w\ldots$

Initialization time: $2^{f(w)} \cdot n \log^2 n$

Update time: $2^{f(w) \cdot \sqrt{\log n \log \log n}}$ *(amortized)*

Extension

We can also dynamically solve any decision/optimization problem expressible in CMSO_1 logic.

Max Clique, Max Independent Set, Min Dominating Set, ~~Longest Path~~...

Dynamic Rankwidth

Rank decomposition algorithms

Given an n-vertex graph G of rankwidth w, we can **find** a rank decomposition of G...

	Width guarantee	**Time**
[Oum, Seymour '06]	$3w + 1$	$2^{\mathcal{O}(w)} \cdot n^9$
[Oum '08]	$3w - 1$	$f(w) \cdot n^3$
[Jeong, Kim, Oum '21]	w	$f(w) \cdot n^3$
[Fomin, Korhonen '22]	w	$f(w) \cdot n^2$

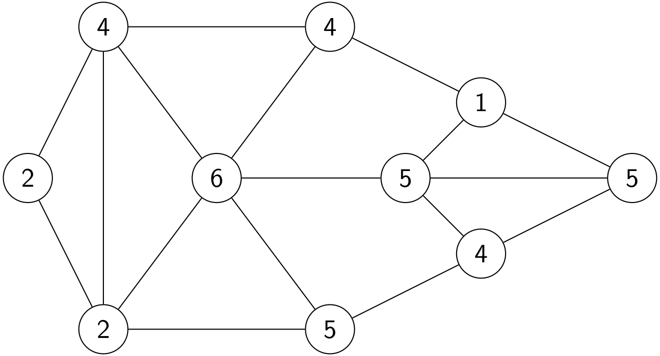
Dynamic Rankwidth

Rank decomposition algorithms

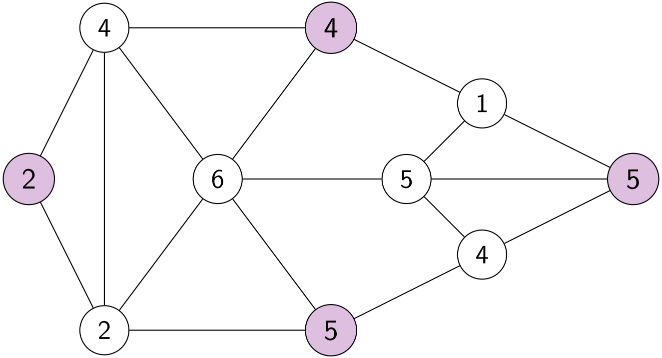
Given an n-vertex graph G of rankwidth w, we can **find** a rank decomposition of G...

	Width guarantee	**Time**
[Oum, Seymour '06]	$3w+1$	$2^{\mathcal{O}(w)} \cdot n^9$
[Oum '08]	$3w-1$	$f(w) \cdot n^3$
[Jeong, Kim, Oum '21]	w	$f(w) \cdot n^3$
[Fomin, Korhonen '22]	w	$f(w) \cdot n^2$
[Korhonen, Sokołowski '24]	$\mathbf{w}$	$\mathbf{f(w) \cdot n^{1+o(1)} + \mathcal{O}(m)}$

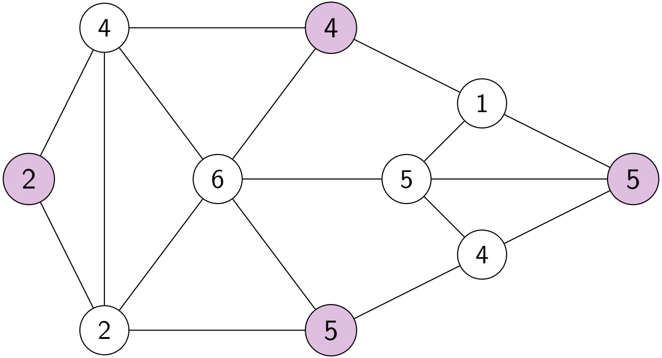
Dynamic Planar Maximum Weighted Independent Set



Dynamic Planar Maximum Weighted Independent Set

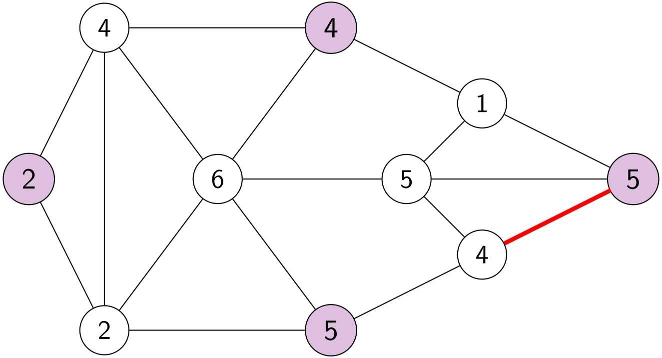


Dynamic Planar Maximum Weighted Independent Set



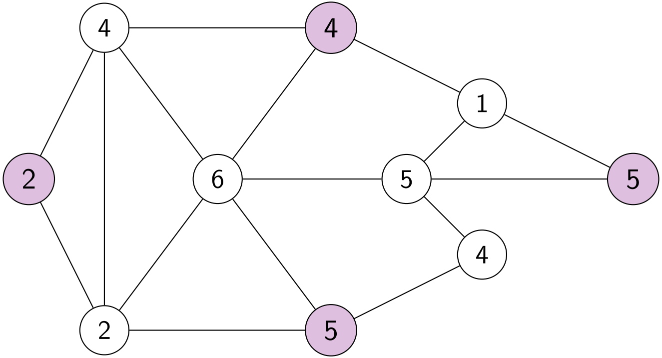
Static variant: NP-hard... but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



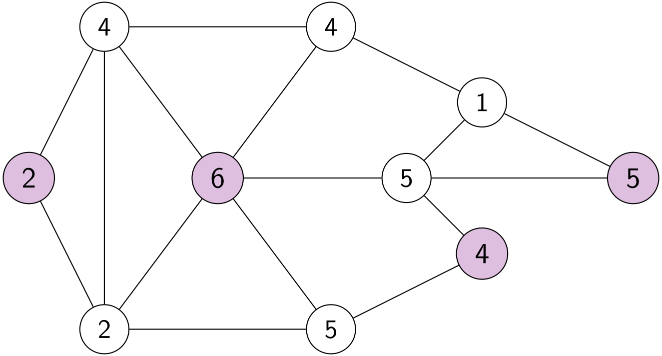
Static variant: NP-hard... but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



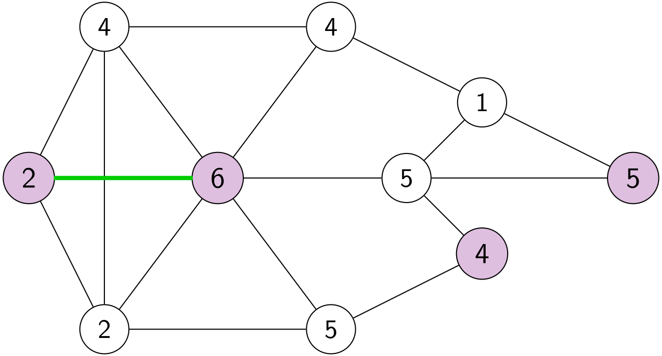
Static variant: NP-hard… but $(1 - \varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



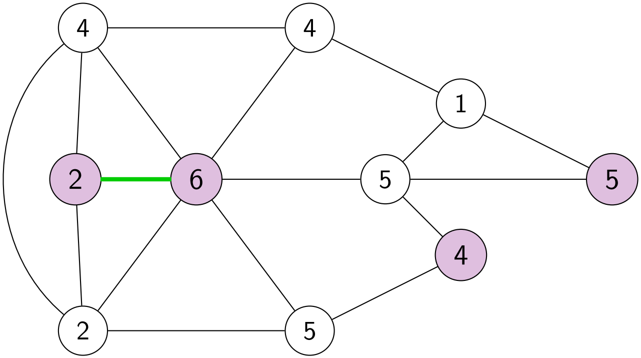
Static variant: NP-hard... but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon)\cdot\mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



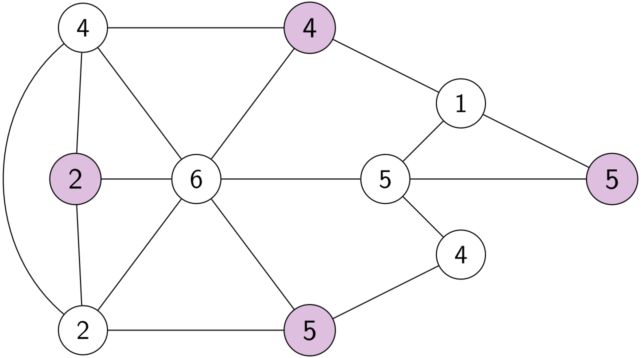
Static variant: NP-hard... but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



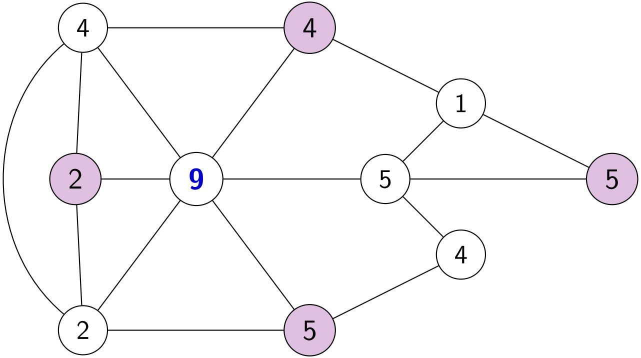
Static variant: NP-hard… but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon)\cdot\mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



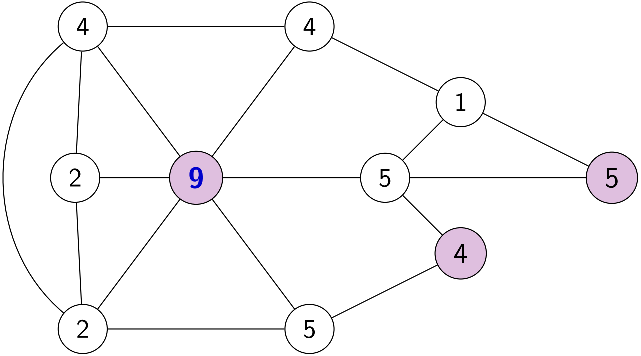
Static variant: NP-hard... but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



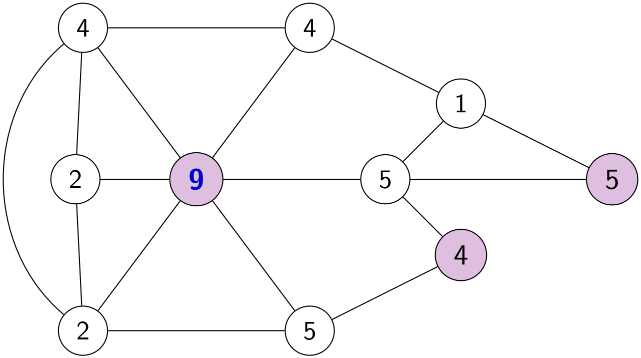
Static variant: NP-hard... but $(1 - \varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



Static variant: NP-hard... but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon) \cdot \mathcal{O}(n)$ [Baker '94]

Dynamic Planar Maximum Weighted Independent Set



Static variant: NP-hard… but $(1-\varepsilon)$-**approximation** in time $f(\varepsilon)\cdot\mathcal{O}(n)$ [Baker '94]

Question: What about **dynamic** approximation schemes?

Dynamic Baker scheme

Korhonen, Nadara, Pilipczuk, **Sokołowski** [SODA '24]

Fully Dynamic Approximation Schemes on Planar and Apex-Minor-Free Graphs

Main result

Dynamic Baker scheme

Korhonen, Nadara, Pilipczuk, **Sokołowski** [SODA '24]

FULLY DYNAMIC APPROXIMATION SCHEMES ON PLANAR AND APEX-MINOR-FREE GRAPHS

Main result

Given: Initially edgeless, vertex-weighted dynamic **planar** graph G.

Dynamic Baker scheme

Korhonen, Nadara, Pilipczuk, **Sokołowski** [SODA '24]

FULLY DYNAMIC APPROXIMATION SCHEMES ON PLANAR AND APEX-MINOR-FREE GRAPHS

Main result

Given: Initially edgeless, vertex-weighted dynamic **planar** graph G. Let also $\varepsilon > 0$.

Dynamic Baker scheme

Korhonen, Nadara, Pilipczuk, **Sokołowski** [SODA '24]

FULLY DYNAMIC APPROXIMATION SCHEMES ON PLANAR AND APEX-MINOR-FREE GRAPHS

Main result

Given: Initially edgeless, vertex-weighted dynamic **planar** graph G. Let also $\varepsilon > 0$.

Then: We can maintain a value $p \geq 0$ so that:

$$(1-\varepsilon)\mathrm{OPT}_{\mathrm{IS}} \leq p \leq \mathrm{OPT}_{\mathrm{IS}},$$

where $\mathrm{OPT}_{\mathrm{IS}}$ is the maximum weight of an independent set in G.

Dynamic Baker scheme

Korhonen, Nadara, Pilipczuk, **Sokołowski** [SODA '24]

FULLY DYNAMIC APPROXIMATION SCHEMES ON PLANAR AND APEX-MINOR-FREE GRAPHS

Main result

Given: Initially edgeless, vertex-weighted dynamic **planar** graph G. Let also $\varepsilon > 0$.

Then: We can maintain a value $p \geq 0$ so that:

$$(1-\varepsilon)\mathrm{OPT}_{\mathrm{IS}} \leq p \leq \mathrm{OPT}_{\mathrm{IS}},$$

where $\mathrm{OPT}_{\mathrm{IS}}$ is the maximum weight of an independent set in G.

Update time: $f(\varepsilon) \cdot n^{o(1)}$ *(amortized)*.

Also generalizations to wider classes of graphs, MIN DOMINATING SET...

Featured works

T. Korhonen, K. Majewski, W. Nadara, Mi. Pilipczuk, M. Sokołowski [FOCS '23]
Dynamic Treewidth

T. Korhonen, M. Sokołowski [STOC '24]
Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

T. Korhonen, W. Nadara, Mi. Pilipczuk, M. Sokołowski [SODA '24]
Fully Dynamic Approximation Schemes on Planar and Apex-Minor-Free Graphs

Mi. Pilipczuk, M. Sokołowski, A. Zych-Pawlewicz [STACS '22]
Compact Representation For Matrices of Bounded Twin-Width

Mi. Pilipczuk, M. Sokołowski [J. Comb. Theory B '24]
Graphs of Bounded Twin-Width Are Quasi-Polynomially χ-Bounded

Featured works

T. Korhonen, K. Majewski, W. Nadara, Mi. Pilipczuk, M. Sokołowski [FOCS '23]
Dynamic Treewidth

T. Korhonen, M. Sokołowski [STOC '24]
Almost-Linear Time Parameterized Algorithm for Rankwidth via Dynamic Rankwidth

T. Korhonen, W. Nadara, Mi. Pilipczuk, M. Sokołowski [SODA '24]
Fully Dynamic Approximation Schemes on Planar and Apex-Minor-Free Graphs

Mi. Pilipczuk, M. Sokołowski, A. Zych-Pawlewicz [STACS '22]
Compact Representation For Matrices of Bounded Twin-Width

Mi. Pilipczuk, M. Sokołowski [J. Comb. Theory B '24]
Graphs of Bounded Twin-Width Are Quasi-Polynomially χ-Bounded

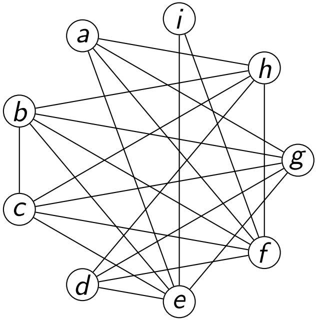
Thank you!

Appendix

Definition of rankwidth

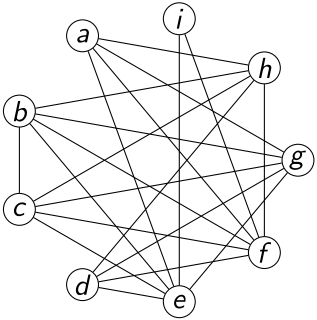
Rankwidth

Aim: recursively vertex-decompose a graph G...



Rankwidth

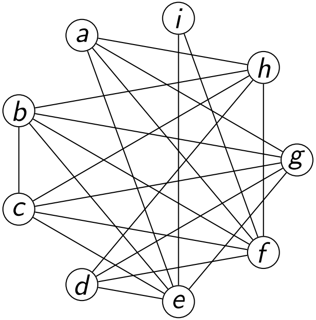
Aim: recursively vertex-decompose a graph G...



$\{a, b, c, d, e, f, g, h, i\}$

Rankwidth

Aim: recursively vertex-decompose a graph G...

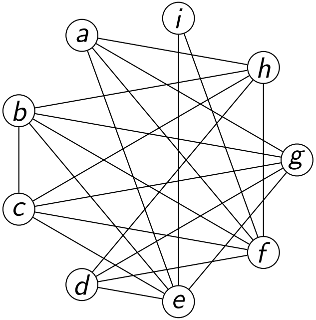


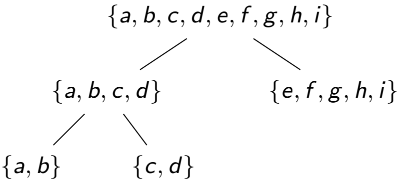
$$\{a, b, c, d, e, f, g, h, i\}$$

$$\{a, b, c, d\} \qquad \{e, f, g, h, i\}$$

Rankwidth

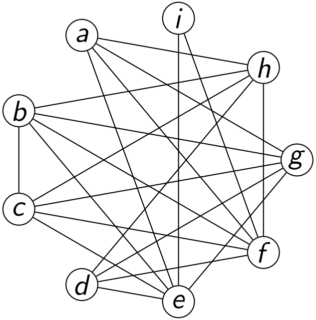
Aim: recursively vertex-decompose a graph G...





Rankwidth

Aim: recursively vertex-decompose a graph G...



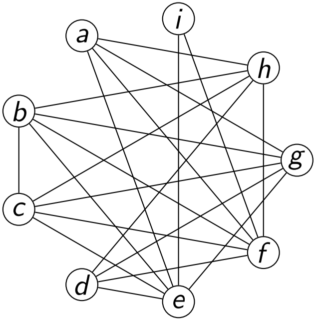
$\{a, b, c, d, e, f, g, h, i\}$

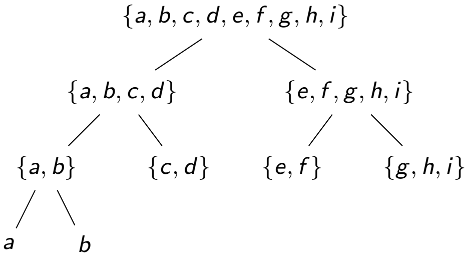
$\{a, b, c, d\}$ $\{e, f, g, h, i\}$

$\{a, b\}$ $\{c, d\}$ $\{e, f\}$ $\{g, h, i\}$

Rankwidth

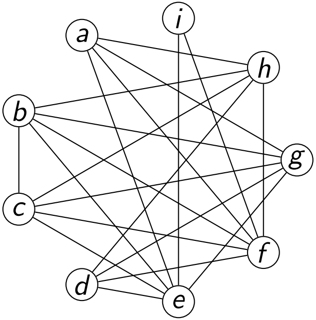
Aim: recursively vertex-decompose a graph G...

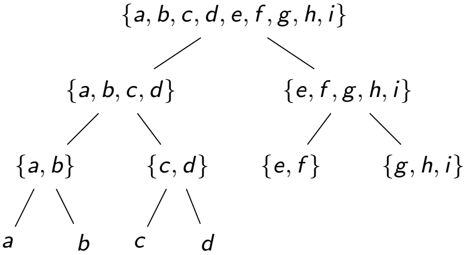




Rankwidth

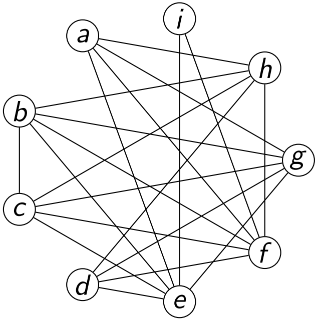
Aim: recursively vertex-decompose a graph G...

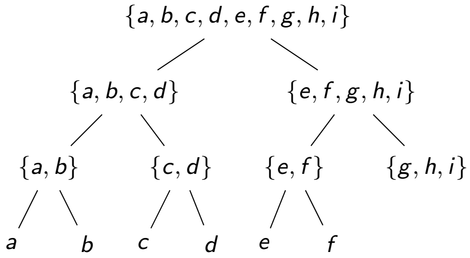




Rankwidth

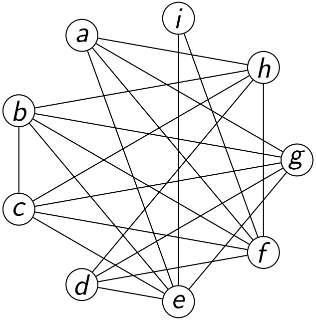
Aim: recursively vertex-decompose a graph G...

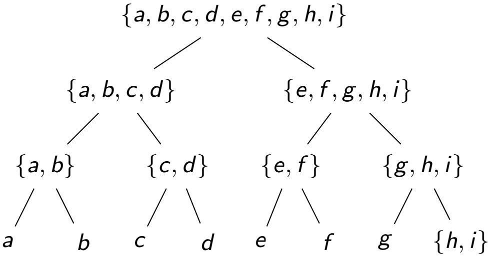




Rankwidth

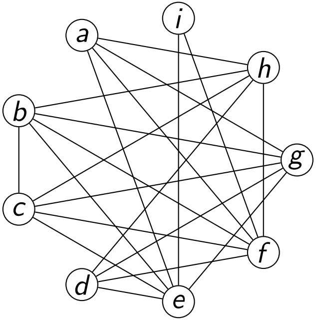
Aim: recursively vertex-decompose a graph G...

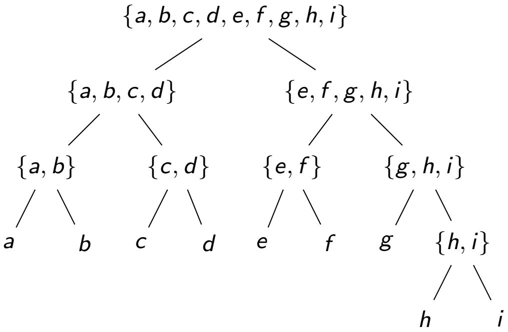




Rankwidth

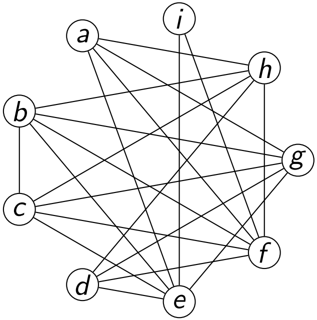
Aim: recursively vertex-decompose a graph G. . .

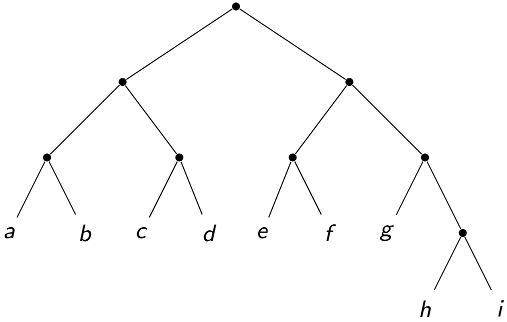




Rankwidth

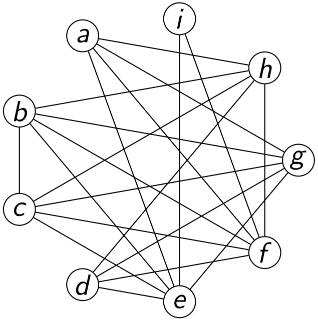
Aim: recursively vertex-decompose a graph G...

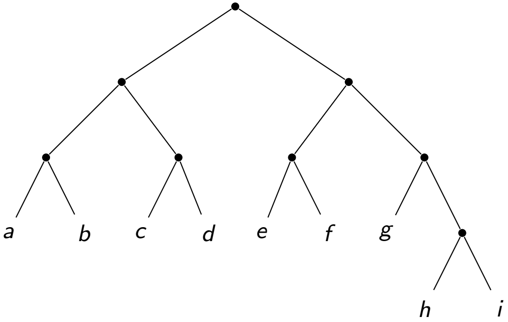




Rankwidth

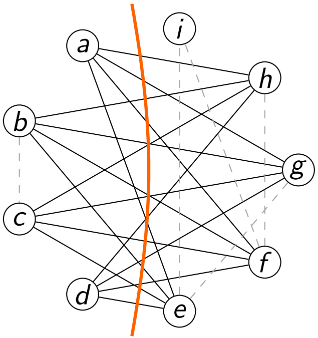
Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**

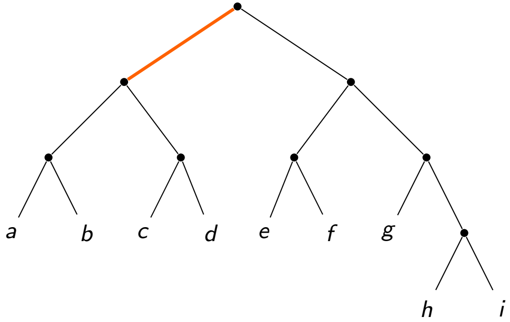




Rankwidth

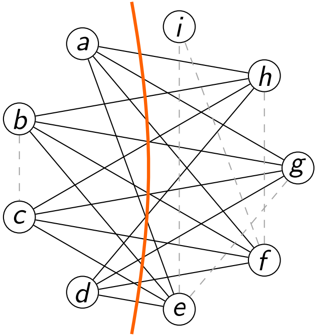
Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**

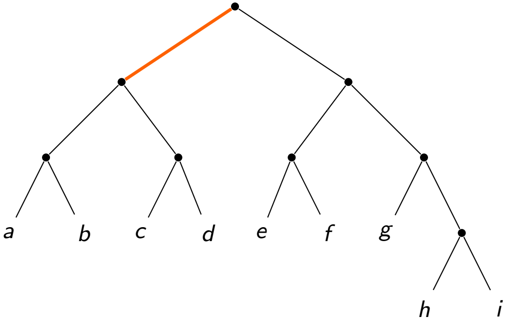




Rankwidth

Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**

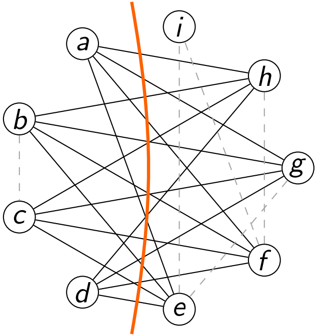




Simple for rankwidth: adjacency matrix of the cut has **small** $\mathrm{GF}(2)$ **rank**

Rankwidth

Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**



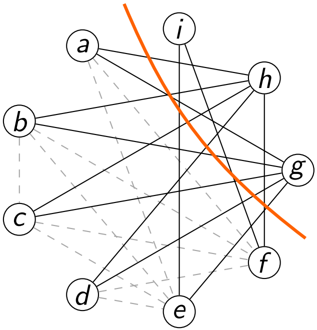
	e	f	g	h	i
a	●	●	●	●	
b	●	●	●	●	
c	●	●	●	●	
d	●	●	●	●	

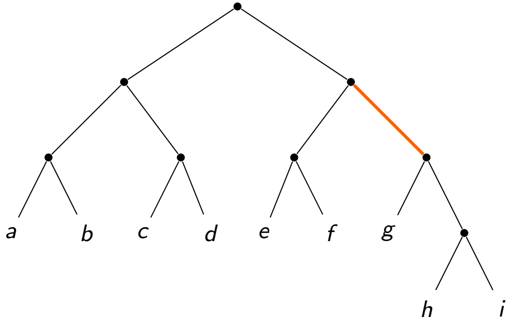
Rank: 1

Simple for rankwidth: adjacency matrix of the cut has **small** $\mathrm{GF}(2)$ **rank**

Rankwidth

Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**

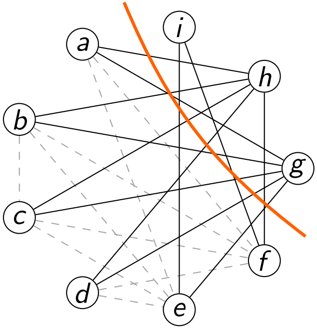




Simple for rankwidth: adjacency matrix of the cut has **small** $\mathrm{GF}(2)$ **rank**

Rankwidth

Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**



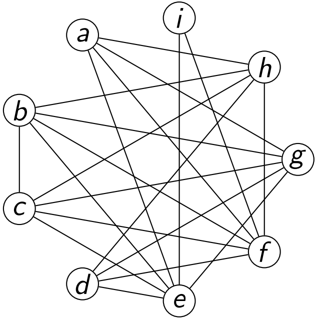
	g	h	i
a	●	●	
b	●	●	
c	●	●	
d	●	●	
e	●		●
f		●	●

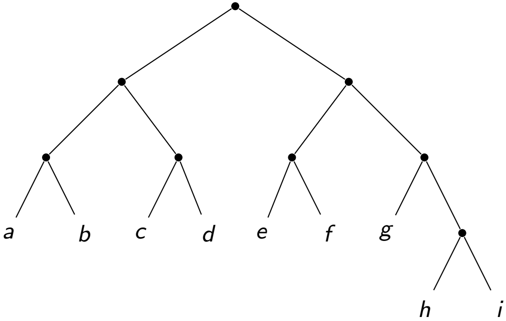
Rank: 2

Simple for rankwidth: adjacency matrix of the cut has **small** $\mathrm{GF}(2)$ **rank**

Rankwidth

Aim: recursively vertex-decompose a graph G… so that each **cut** is **simple**

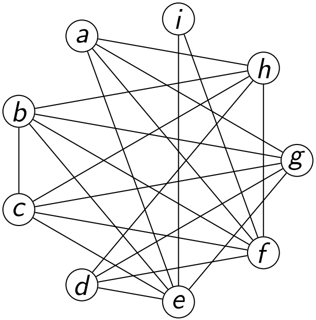


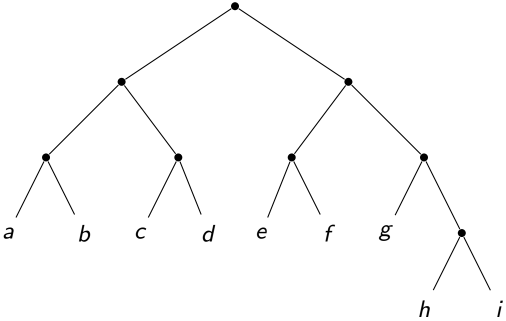


Width of a cut $(X, \overline{X})$: $\mathrm{GF}(2)$ rank of the adjacency matrix of $G[X, \overline{X}]$

Rankwidth

Aim: recursively vertex-decompose a graph $G\dots$ so that each **cut** is **simple**



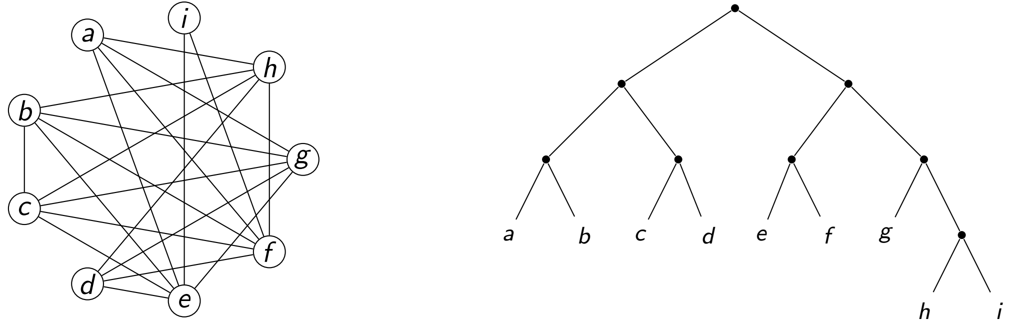


Width of a cut $(X, \overline{X})$: $\mathrm{GF}(2)$ rank of the adjacency matrix of $G[X, \overline{X}]$

Width of a **rank decomposition** $\mathcal{T}$ of G: maximum width of a cut given by $\mathcal{T}$

Rankwidth

Aim: recursively vertex-decompose a graph G... so that each **cut** is **simple**



Width of a cut $(X, \overline{X})$: $\mathrm{GF}(2)$ rank of the adjacency matrix of $G[X, \overline{X}]$

Width of a **rank decomposition** $\mathcal{T}$ of G: maximum width of a cut given by $\mathcal{T}$

Rankwidth of G: minimum width of a rank decomposition of G