Kernel Bounds for Structural Parameterizations of Pathwidth

Bart M. P. Jansen

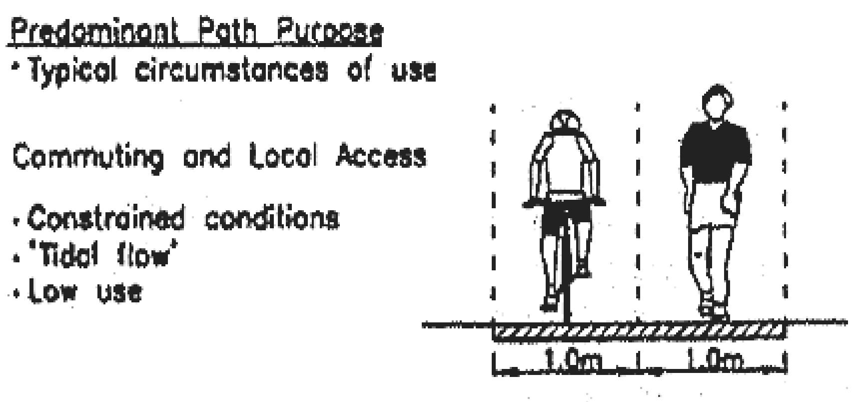
Joint work with
Hans L. Bodlaender & Stefan Kratsch

Universiteit Utrecht

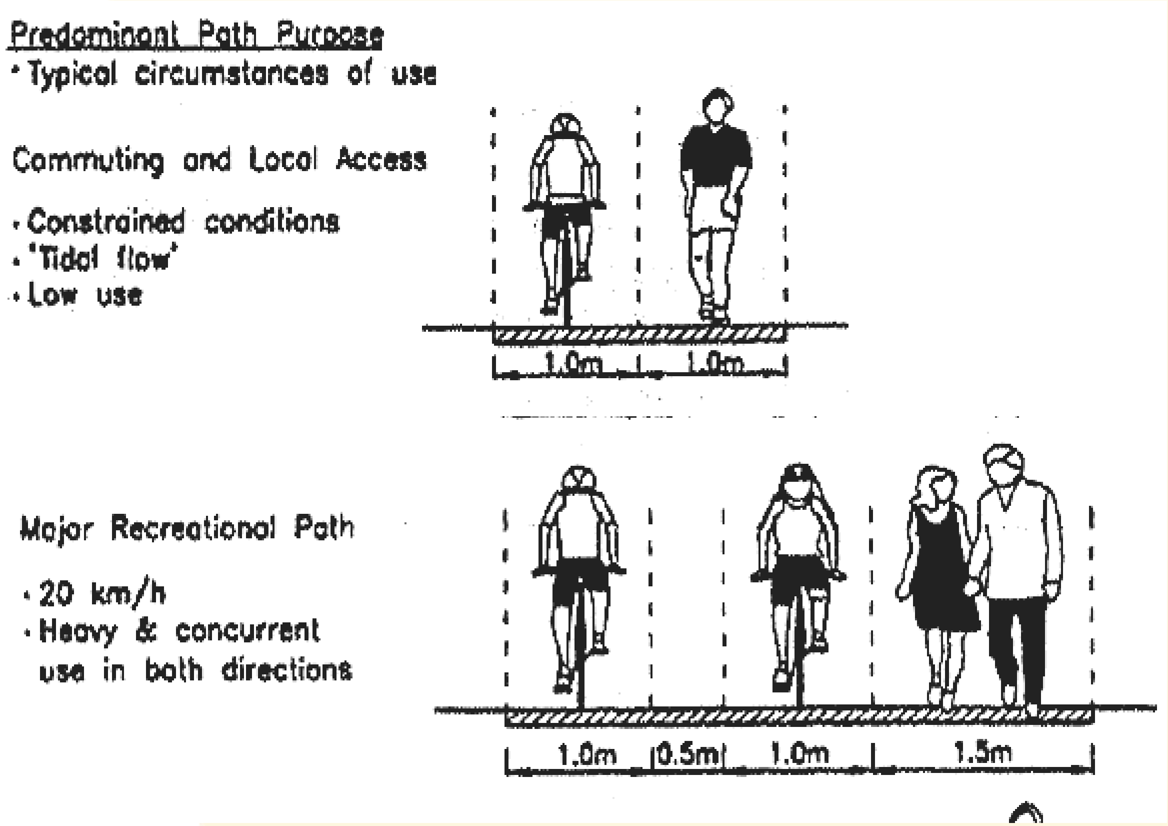
July 6th 2012, SWAT 2012, Helsinki

What is pathwidth?

What is pathwidth?



What is pathwidth?



What is pathwidth?

<u>Predominant Path Purpose</u>
• Typical circumstances of us

Commuting and Local Acces

• Constrained conditions
• 'Tidal flow'
• Low use

Major Recreational Path

• 20 km/h
• Heavy & concurrent use in both directions

What is pathwidth?

- Measure of how “path-like” a graph is
  - Related to treewidth (“tree-like”)

What is pathwidth?

- Measure of how "path-like" a graph is
  - Related to treewidth ("tree-like")
- Gives the quality of a **path decomposition**, a decomposition of a graph into pieces arranged on a path

What is pathwidth?

- Measure of how "path-like" a graph is
  - Related to treewidth ("tree-like")
- Gives the quality of a **path decomposition**, a decomposition of a graph into pieces arranged on a path
- Both pathwidth and treewidth have been introduced many times under different names
  - (vertex separation number, node search number, partial k-tree, etc …)

What is pathwidth?

- Measure of how "path-like" a graph is
  - Related to treewidth ("tree-like")
- Gives the quality of a **path decomposition**, a decomposition of a graph into pieces arranged on a path
- Both pathwidth and treewidth have been introduced many times under different names
  - (vertex separation number, node search number, partial k-tree, etc …)
- Play crucial roles in Robertson & Seymour's proof of the Graph Minor Theorem

Why is it important?

- Many graph problems can be solved efficiently (in linear time) if a path or tree decomposition of small width is known

Why is it important?

- Many graph problems can be solved efficiently (in linear time) if a path or tree decomposition of small width is known
- When comparing pathwidth to treewidth:
  - Path decompositions have larger width
  - Dynamic programming algorithms for path decompositions are simpler and use less memory

Why is it important?

- Many graph problems can be solved efficiently (in linear time) if a path or tree decomposition of small width is known
- When comparing pathwidth to treewidth:
  - Path decompositions have larger width
  - Dynamic programming algorithms for path decompositions are simpler and use less memory
- Important to find **low-width** path and tree decompositions **efficiently**

Finding good decompositions is hard

- Computing pathwidth or treewidth of a graph is NP-complete
  - Pathwidth is even NP-complete on planar graphs
  - Treewidth of planar graphs is open

Finding good decompositions is hard

- Computing pathwidth or treewidth of a graph is NP-complete
  - Pathwidth is even NP-complete on planar graphs
  - Treewidth of planar graphs is open
- No constant-factor approximation algorithms known
  - Use heuristics, or exponential-time algorithms

Finding good decompositions is hard

- Computing pathwidth or treewidth of a graph is NP-complete
  - Pathwidth is even NP-complete on planar graphs
  - Treewidth of planar graphs is open
- No constant-factor approximation algorithms known
  - Use heuristics, or exponential-time algorithms
- There are $2^{poly(k)}$ n algorithms that either:
  - Compute a decomposition of width k
  - Determine that no such decomposition exists

Finding good decompositions is hard

- Computing pathwidth or treewidth of a graph is NP-complete
  - Pathwidth is even NP-complete on planar graphs
  - Treewidth of planar graphs is open
- No constant-factor approximation algorithms known
  - Use heuristics, or exponential-time algorithms
- There are $2^{poly(k)}$ n algorithms that either:
  - Compute a decomposition of width k
  - Determine that no such decomposition exists

- Runtime ☹(n) for every fixed k

Preprocessing

- Preprocess G to find a smaller graph G', such that:
  - Path decomposition of G' can be lifted efficiently to decomposition of G
  - Lifting does not increase the width

Preprocessing

- Preprocess G to find a smaller graph G′, such that:
  - Path decomposition of G′ can be lifted efficiently to decomposition of G
  - Lifting does not increase the width
- After preprocessing, find a decomposition for G′ by an exponential-time algorithm or heuristics

Preprocessing

- Preprocess G to find a smaller graph G′, such that:
  - Path decomposition of G′ can be lifted efficiently to decomposition of G
  - Lifting does not increase the width
- After preprocessing, find a decomposition for G′ by an exponential-time algorithm or heuristics
- We want to give a guarantee on the size of the output
  - Kernelization

Preprocessing

- Preprocess G to find a smaller graph G', such that:
  - Path decomposition of G' can be lifted efficiently to decomposition of G
  - Lifting does not increase the width
- After preprocessing, find a decomposition for G' by an exponential-time algorithm or heuristics
- We want to give a guarantee on the size of the output
  - Kernelization
- Cannot guarantee output is smaller than input (else P=NP)

Preprocessing

- Preprocess G to find a smaller graph G′, such that:
  - Path decomposition of G′ can be lifted efficiently to decomposition of G
  - Lifting does not increase the width
- After preprocessing, find a decomposition for G′ by an exponential-time algorithm or heuristics
- We want to give a guarantee on the size of the output
  - Kernelization
- Cannot guarantee output is smaller than input (else P=NP)
- So given a graph G of “difficulty” k, shrink G to poly(k)
  - Afterwards we can shrink no more

Setting realistic goals

- Cannot preprocess G to size poly(pw(G)) without changing the pathwidth
  - Unless NP $\subseteq$ coNP/poly [BDFH'08,D'12]
  - k-Pathwidth is AND-compositional

Setting realistic goals

- Cannot preprocess G to size poly(pw(G)) without changing the pathwidth
  - Unless NP ⊆ coNP/poly [BDFH'08,D'12]
  - k-Pathwidth is AND-compositional
- Pick a measure for graph difficulty that is larger than pw(G)

Setting realistic goals

- Cannot preprocess G to size poly(pw(G)) without changing the pathwidth
  - Unless NP ⊆ coNP/poly [BDFH’08,D’12]
  - k-Pathwidth is AND-compositional
- Pick a measure for graph difficulty that is larger than pw(G)
- Can we shrink to size polynomial in the larger measure?
  - For example: size of a minimum vertex cover
  - (Vertex set that covers all edges)

The preprocessing story so far …

The preprocessing story so far …

- Lots of work on preprocessing for **treewidth**

The preprocessing story so far …

- Lots of work on preprocessing for **treewidth**
- Heuristic reduction rules with experimental evaluations

The preprocessing story so far …

- Lots of work on preprocessing for **treewidth**
- Heuristic reduction rules with experimental evaluations
- Rules were found to work well in practice
  - No theoretical justification

The preprocessing story so far …

- Lots of work on preprocessing for **treewidth**
- Heuristic reduction rules with experimental evaluations
- Rules were found to work well in practice
  - No theoretical justification
- BJK '11:
  - Existing reduction rules give size reduction to $O(VC^3)$
  - With some more rules, size reduction to $O(FVS^4)$
  - Heuristic rules have **provable** effect!

The preprocessing story so far …

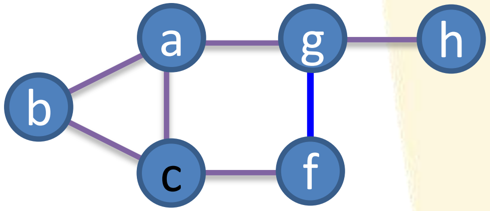
- Lots of work on preprocessing for **treewidth**
- Heuristic reduction rules with experimental evaluations
- Rules were found to work well in practice
  - No theoretical justification
- BJK '11:
  - Existing reduction rules give size reduction to $O(VC^3)$
  - With some more rules, size reduction to $O(FVS^4)$
  - Heuristic rules have **provable** effect!
- No prior work on preprocessing for pathwidth

The preprocessing story so far …

- Lots of work on preprocessing for **treewidth**
- Heuristic reduction rules with experimental evaluations
- Rules were found to work well in practice
  - No theoretical justification
- BJK ‘11:
  - Existing reduction rules give size reduction to $O(VC^3)$
  - With some more rules, size reduction to $O(FVS^4)$
  - Heuristic rules have **provable** effect!
- No prior work on preprocessing for pathwidth
- This work: reduction rules, analysis & lower bounds

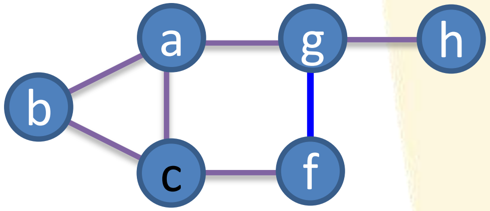
Path decomposition

Path decomposition



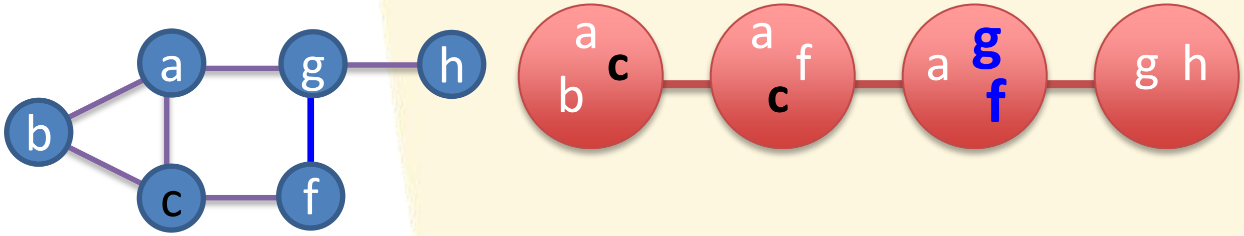
Path decomposition

- A path decomposition of a graph $G=(V,E)$ is a sequence $(X_1, \ldots, X_r)$ of subsets of V, called **bags**, such that:



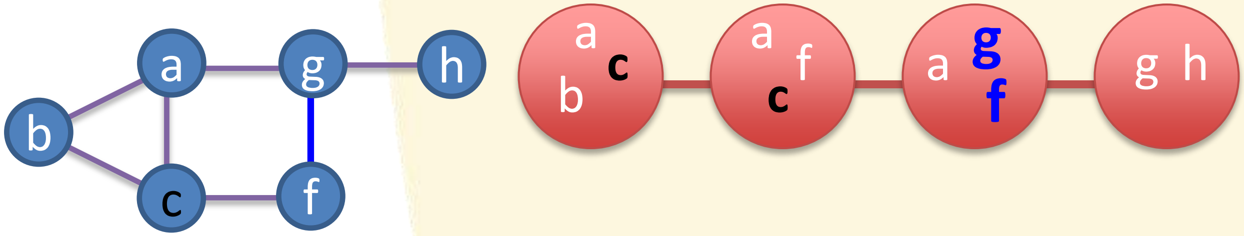
Path decomposition

- A path decomposition of a graph G=(V,E) is a sequence $(X_1, \ldots, X_r)$ of subsets of V, called **bags**, such that:



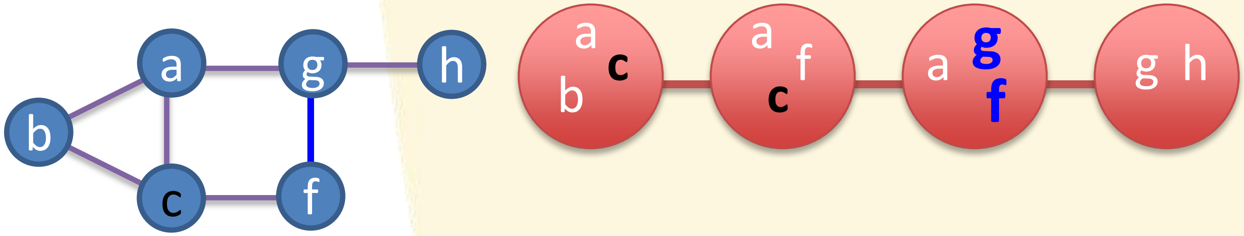
Path decomposition

- A path decomposition of a graph G=(V,E) is a sequence $(X_1, \ldots, X_r)$ of subsets of V, called **bags**, such that:
  – For all *v*, there is a bag that contains *v*



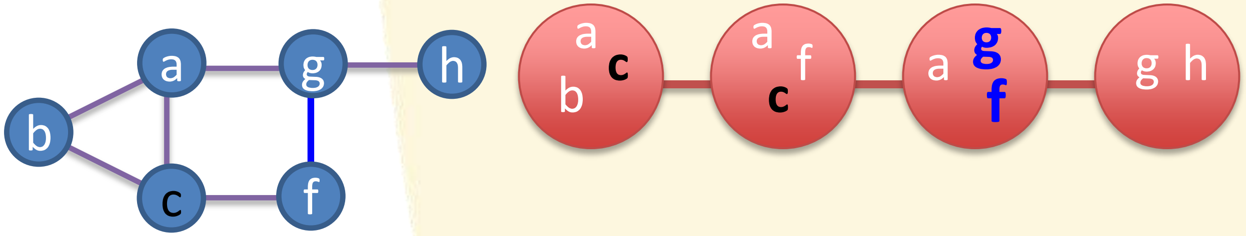
Path decomposition

- A path decomposition of a graph G=(V,E) is a sequence $(X_1, \ldots, X_r)$ of subsets of V, called **bags**, such that:
  - For all *v*, there is a bag that contains *v*
  - For all $\{v,w\} \in E$, there is a bag that contains *v* and *w*



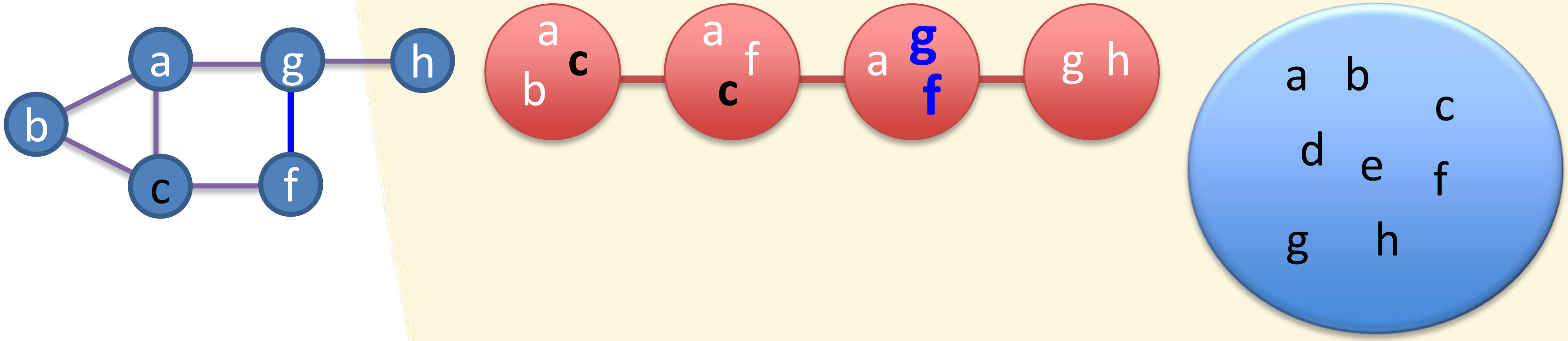
Path decomposition

- A path decomposition of a graph G=(V,E) is a sequence $(X_1, \ldots, X_r)$ of subsets of V, called **bags**, such that:
  - For all *v*, there is a bag that contains *v*
  - For all $\{v,w\} \in E$, there is a bag that contains *v* and *w*
  - For all *v*, the bags that contain *v* are consecutive

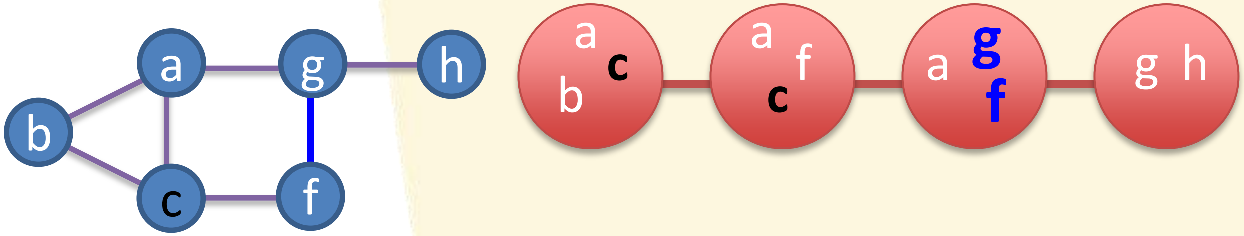


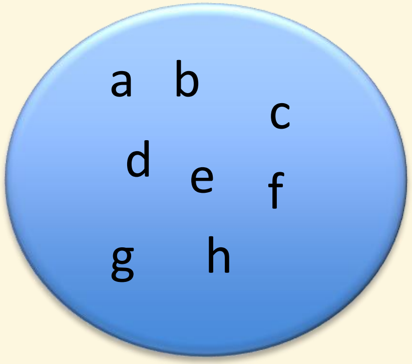
Path decomposition

- A path decomposition of a graph G=(V,E) is a sequence $(X_1, \ldots, X_r)$ of subsets of V, called **bags**, such that:
  - For all *v*, there is a bag that contains *v*
  - For all $\{v,w\} \in E$, there is a bag that contains *v* and *w*
  - For all *v*, the bags that contain *v* are consecutive



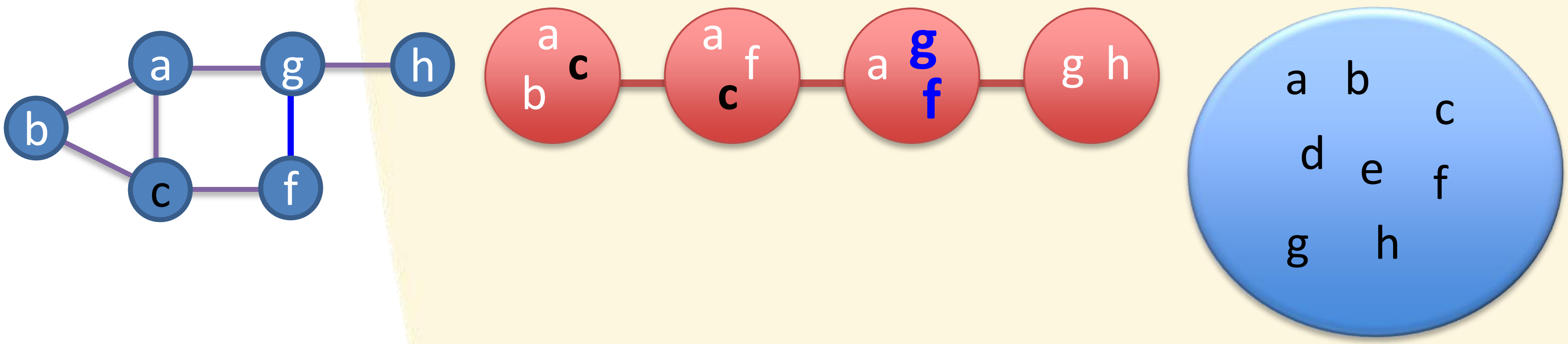
Path decomposition





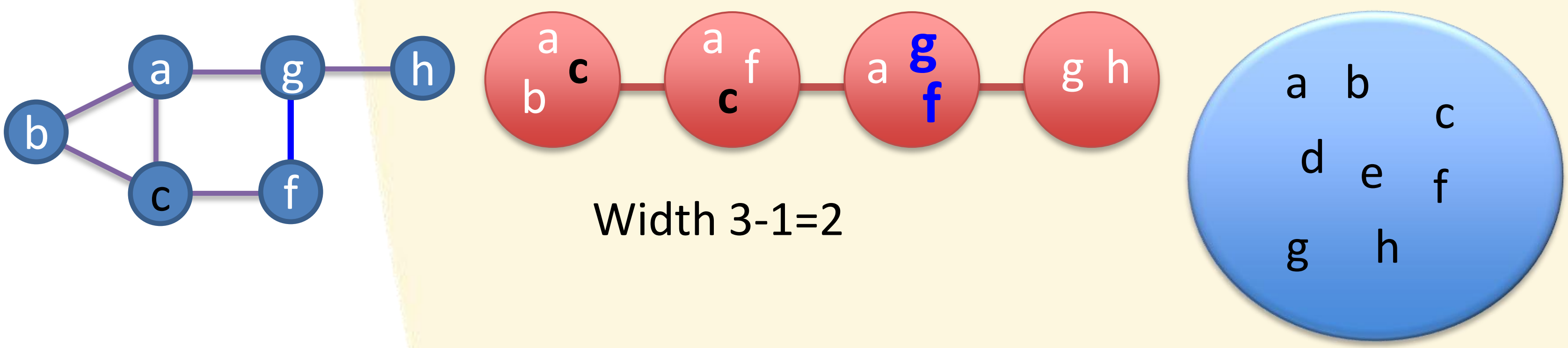
Path decomposition

- The *width* of a path decomposition (X_1, …, X_r) is the size of its largest bag minus one: $\max_{1 \le i \le r} |X_i|-1$



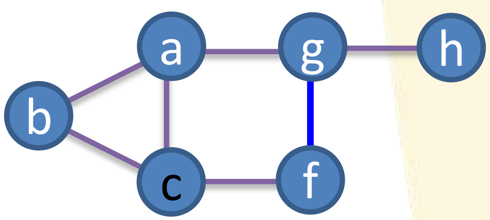
Path decomposition

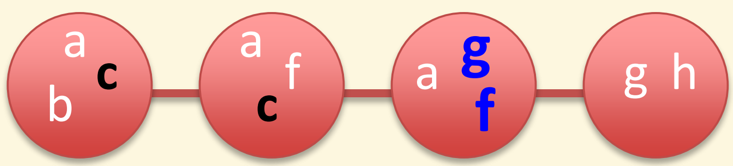
- The *width* of a path decomposition $(X_1, \ldots, X_r)$ is the size of its largest bag minus one: $\max_{1 \leq i \leq r} |X_i| - 1$



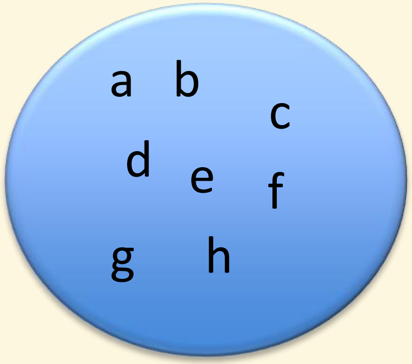
Path decomposition

- The *width* of a path decomposition (X_1, …, X_r) is the size of its largest bag minus one: $\max_{1 \leq i \leq r} |X_i|-1$
- The *pathwidth* of a graph G is the minimum width of a path decomposition of G



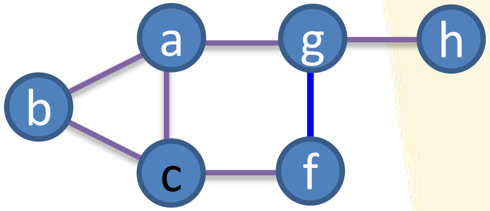


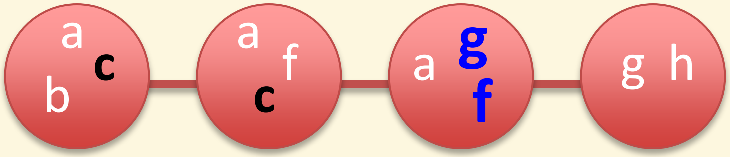
Width 3-1=2



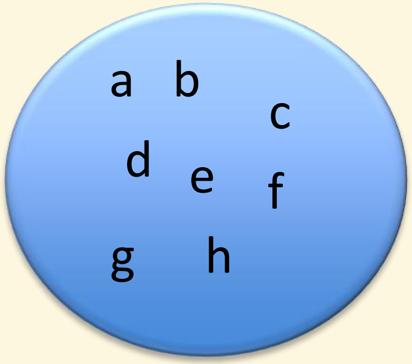
Path decomposition

- Path/treewidth does not increase when deleting or contracting edges / vertices



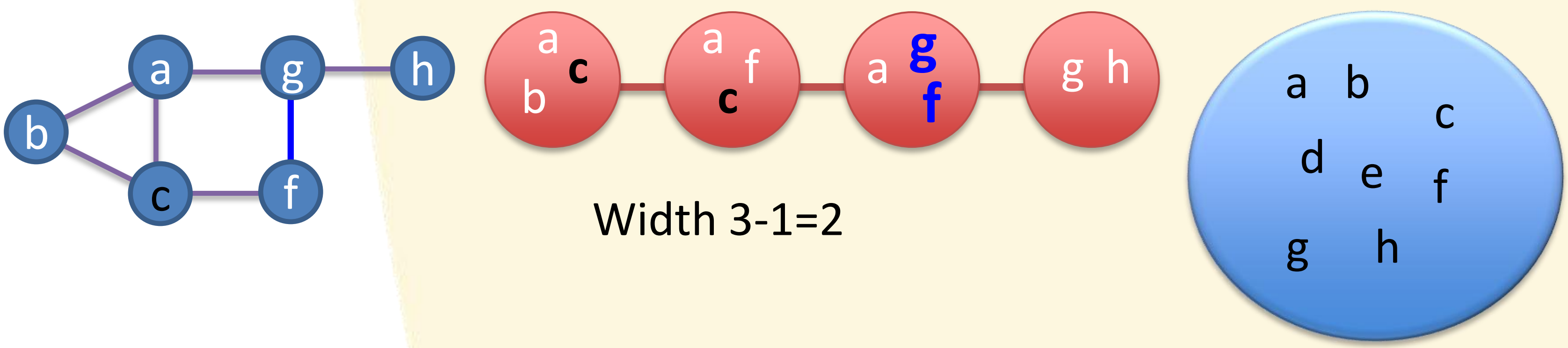


Width 3-1=2



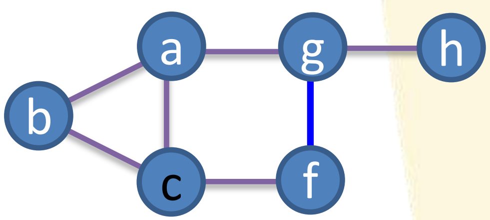
Path decomposition

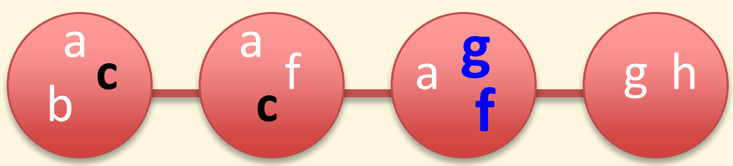
- Path/treewidth does not increase when deleting or contracting edges / vertices
- Treewidth ≤ pathwidth



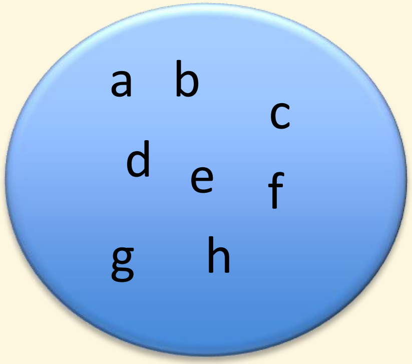
Path decomposition

- Path/treewidth does not increase when deleting or contracting edges / vertices
- Treewidth ≤ pathwidth
- Paths have pathwidth = treewidth = 1



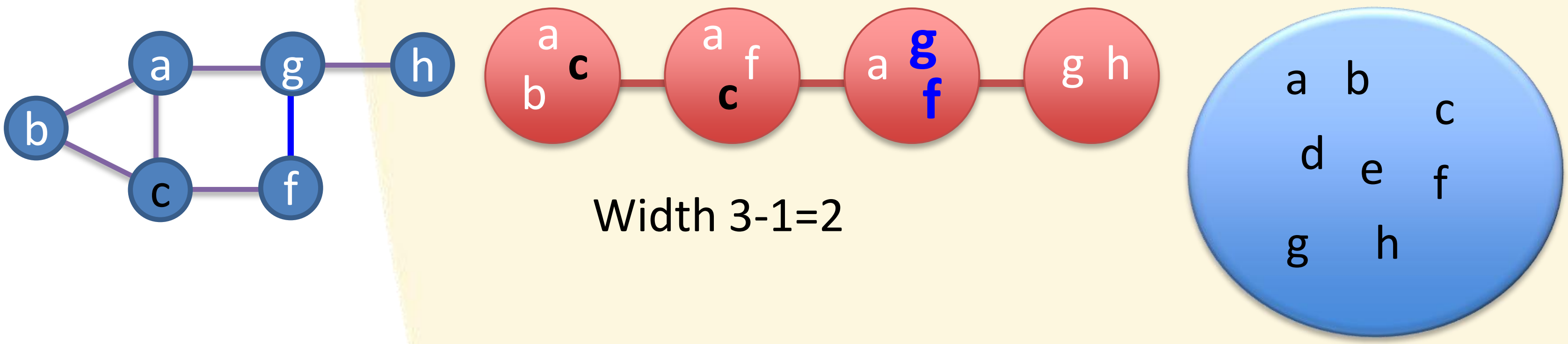


Width 3-1=2



Path decomposition

- Path/treewidth does not increase when deleting or contracting edges / vertices
- Treewidth ≤ pathwidth
- Paths have pathwidth = treewidth = 1
- Trees have treewidth 1, but may have pathwidth $\Theta(\log n)$



Problem setting

- Decision problem associated to pathwidth
  - **Instance**: Graph G, integer k.
  - **Question**: Does G have pathwidth ≤ k?

Problem setting

- Decision problem associated to pathwidth
  - **Instance**: Graph G, integer k.
  - **Question**: Does G have pathwidth ≤ k?

- A reduction from G to G′ is **safe for pathwidth k** if it preserves whether the graph has pathwidth ≤ k
  - (G has pathwidth ≤ k) iff (G′ has pathwidth ≤ k)

Problem setting

- Decision problem associated to pathwidth
  - **Instance**: Graph G, integer k.
  - **Question**: Does G have pathwidth $\leq$ k?

- A reduction from G to G′ is **safe for pathwidth k** if it preserves whether the graph has pathwidth $\leq$ k
  - (G has pathwidth $\leq$ k) iff (G′ has pathwidth $\leq$ k)

- Easy to lift decompositions of G′ to G

Problem setting

- Decision problem associated to pathwidth
  - **Instance**: Graph G, integer k.
  - **Question**: Does G have pathwidth ≤ k?

- A reduction from G to G′ is **safe for pathwidth k** if it preserves whether the graph has pathwidth ≤ k
  - (G has pathwidth ≤ k) iff (G′ has pathwidth ≤ k)

- Easy to lift decompositions of G′ to G
- In practical settings:
  - Guess k, or work with upper- and lower bounds

Common neighbors

Common neighbors

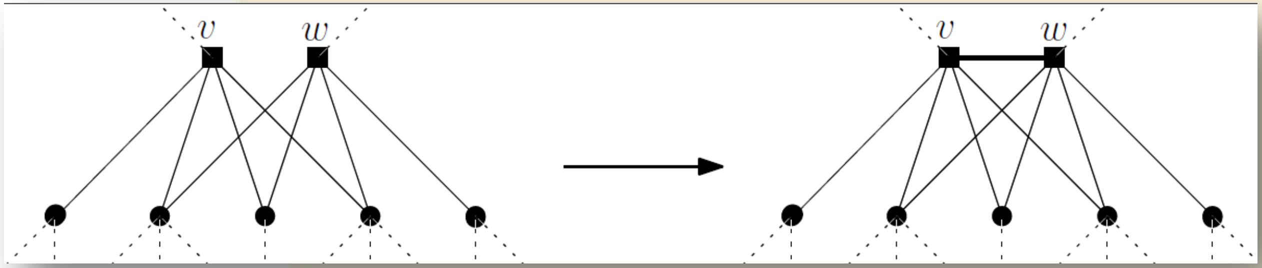
Treewidth Edge Improvement Rule

If v and w have $\geq k+1$ common neighbors in G, then adding edge $\{v,w\}$ does not change whether $tw(G) \leq k$

Common neighbors

Treewidth Edge Improvement Rule

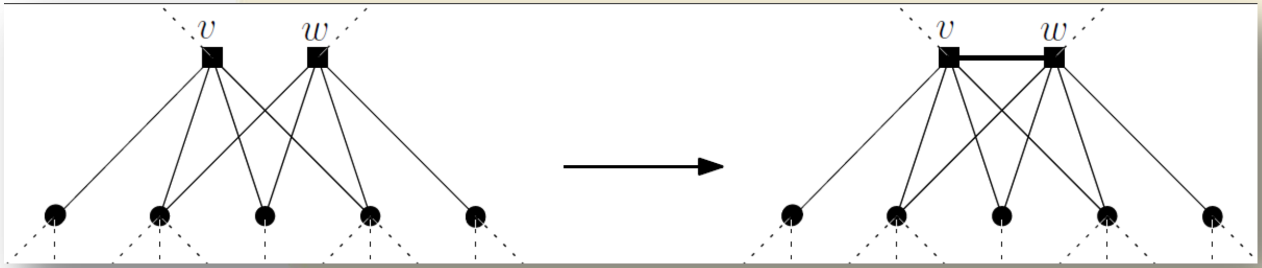
If v and w have ≥ k+1 common neighbors in G, then adding edge {v,w} does not change whether tw(G) ≤ k



Common neighbors

- Rule originates from Bodlaender's linear-time algorithm for Treewidth

Treewidth Edge Improvement Rule
If v and w have ≥ k+1 common neighbors in G, then adding edge {v,w} does not change whether tw(G) ≤ k

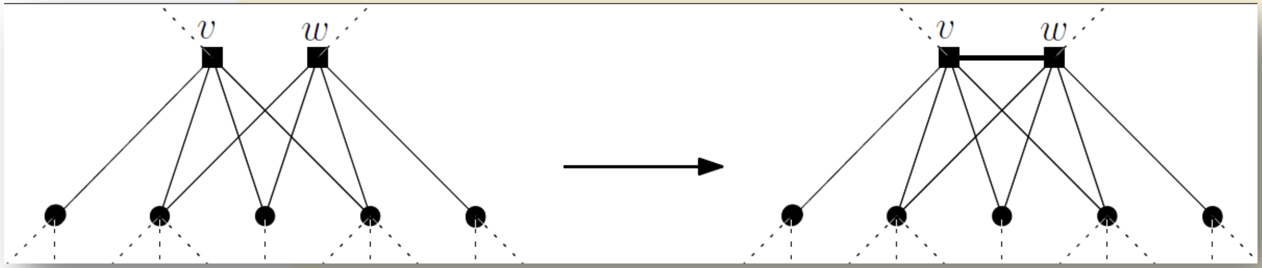


Common neighbors

- Rule originates from Bodlaender's linear-time algorithm for Treewidth
- Any width-k **tree** decomposition has a bag with v and w
  - Hence any width-k **path** decomposition has a {v,w} bag

Treewidth Edge Improvement Rule

If v and w have ≥ k+1 common neighbors in G, then adding edge {v,w} does not change whether tw(G) ≤ k

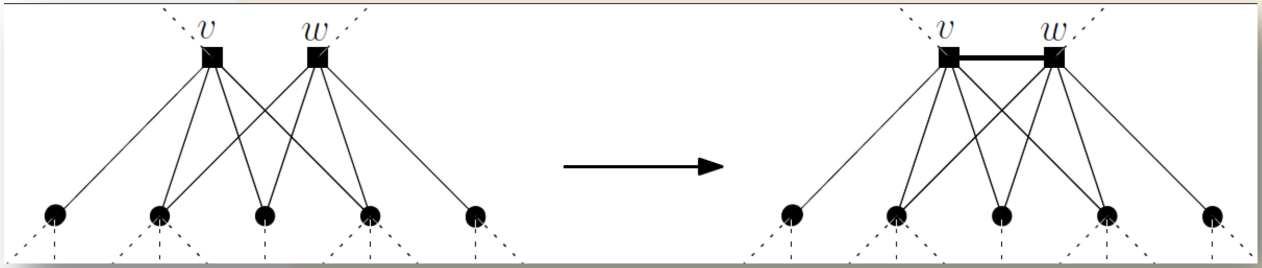


Common neighbors

- Rule originates from Bodlaender's linear-time algorithm for Treewidth
- Any width-k **tree** decomposition has a bag with v and w
  - Hence any width-k **path** decomposition has a {v,w} bag

Pathwidth Edge Improvement Rule

If v and w have ≥ k+1 common neighbors in G, then adding edge {v,w} does not change whether pw(G) ≤ k



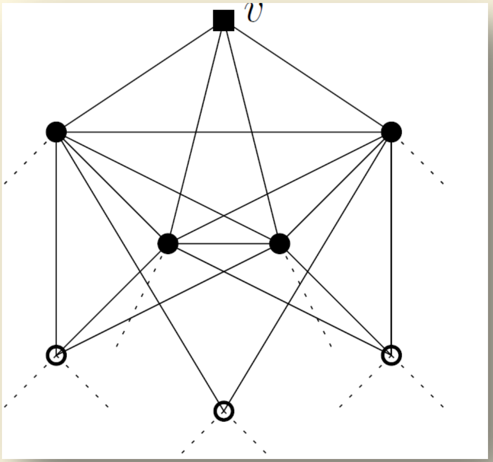
Simplicial Vertices

Simplicial Vertices

- A vertex v is simplicial if N(v) is a clique

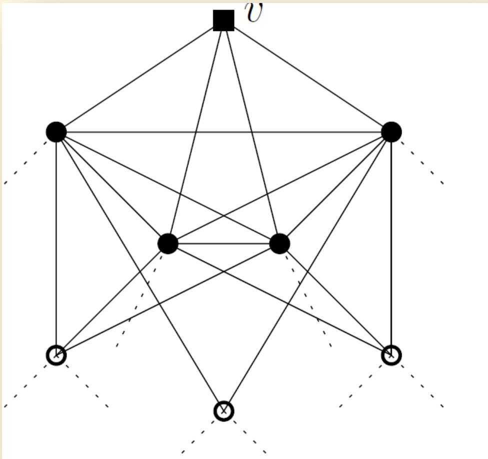
Simplicial Vertices

- A vertex v is simplicial if N(v) is a clique



Simplicial Vertices

- A vertex v is simplicial if N(v) is a clique



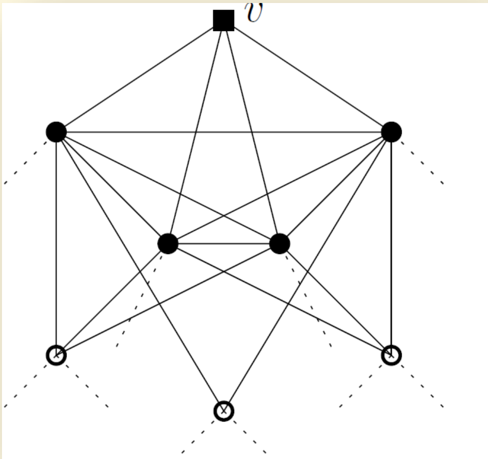
Treewidth Simplicial Vertex Rule

Let v be a simplicial vertex in G.

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Simplicial Vertices

- A vertex v is simplicial if N(v) is a clique



Closed neighborhood of v is a k+2 clique

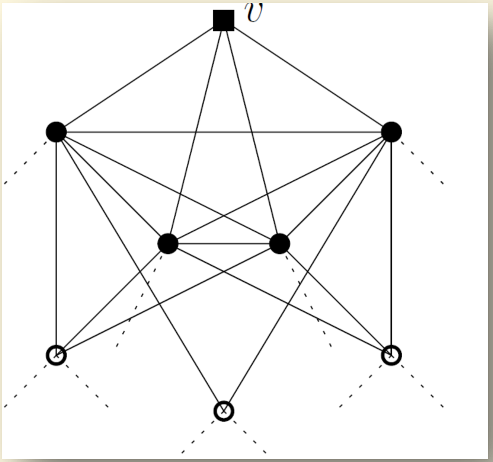
Treewidth Simplicial Vertex Ru

Let v be a simplicial vertex in G.

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Simplicial Vertices

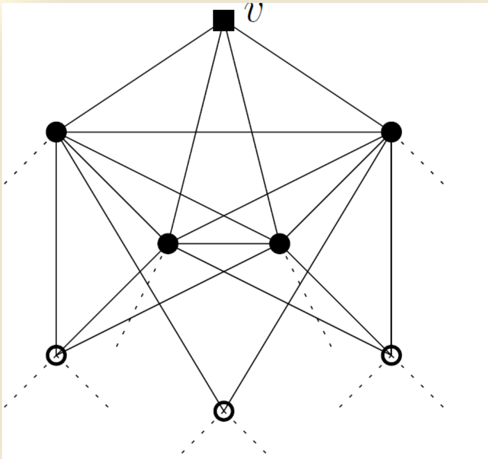
- A vertex v is simplicial if N(v) is a clique



Treewidth Simplicial Vertex Rule
Let v be a simplicial vertex in G.
- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Simplicial Vertices

- A vertex v is simplicial if N(v) is a clique



Treewidth Simplicial Vertex Ru

Let v be a simplicial vertex in G

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Helly property for trees ensures that there is a bag containing N(v)

Simplicial Vertices

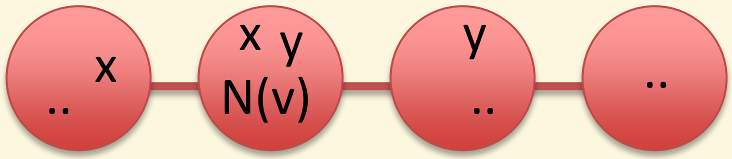
Treewidth Simplicial Vertex Ru

Let v be a simplicial vertex in G

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Helly property for trees ensures that there is a bag containing N(v)

Simplicial Vertices



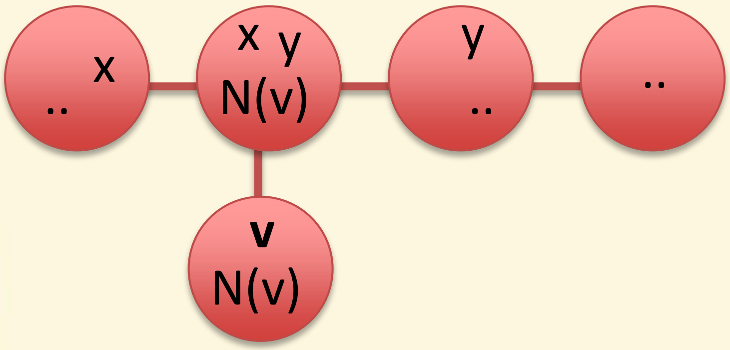
Treewidth Simplicial Vertex Ru

Let v be a simplicial vertex in G

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Helly property for trees ensures that there is a bag containing N(v)

Simplicial Vertices



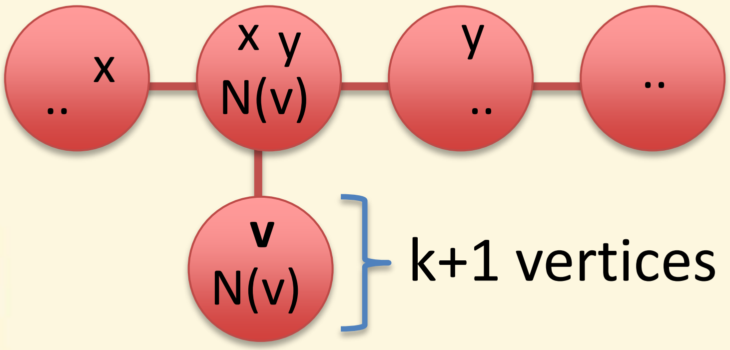
Helly property for trees ensures that there is a bag containing N(v)

Treewidth Simplicial Vertex Ru

Let v be a simplicial vertex in G

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Simplicial Vertices



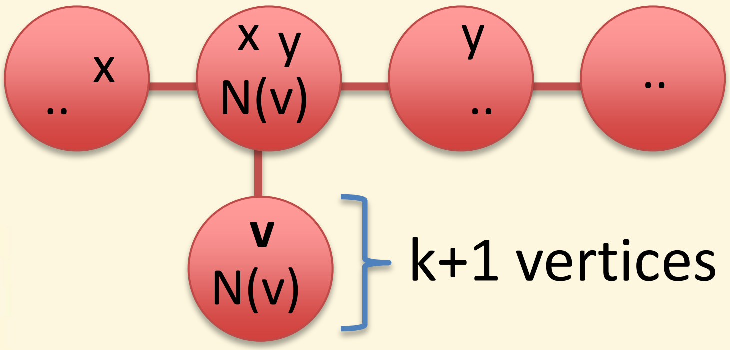
Treewidth Simplicial Vertex Ru

Let v be a simplicial vertex in G

- If deg(v) ≥ k+1 then tw(G) > k
- If deg(v) ≤ k then deleting v is safe for treewidth k

Helly property for trees ensures that there is a bag containing N(v)

Simplicial Vertices

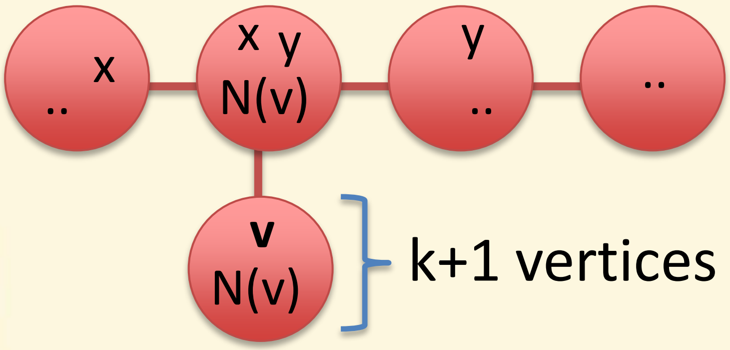


Pathwidth Simplicial Vertex Rule?

Let v be a simplicial vertex in G.

- If deg(v) ≥ k+1 then pw(G) > k
- If deg(v) ≤ k then deleting v is safe for pathwidth k

Simplicial Vertices

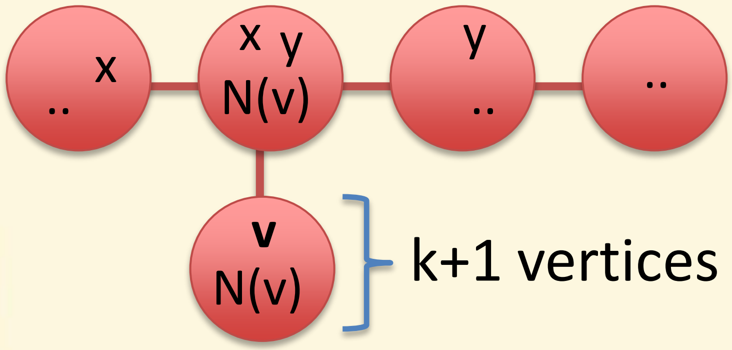


Pathwidth Simplicial Vertex Rule?

Let v be a simplicial vertex in G.

- If deg(v) ≥ k+1 then pw(G) > k
- ~~If deg(v) ≤ k then deleting v is safe for pathwidth k~~

Simplicial Vertices



Pathwidth Simplicial Vertex Rul

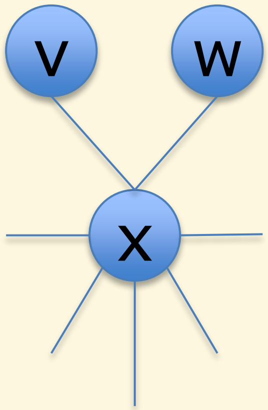
Let v be a simplicial vertex in G.

- If deg(v) ≥ k+1 then pw(G) > k
- ~~If deg(v) ≤ k then deleting v is safe for pathwidth k~~

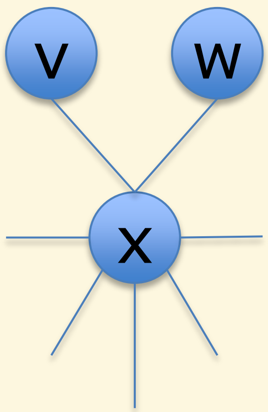
Repeated application would eat up a tree

Degree-one vertices

Degree-one vertices

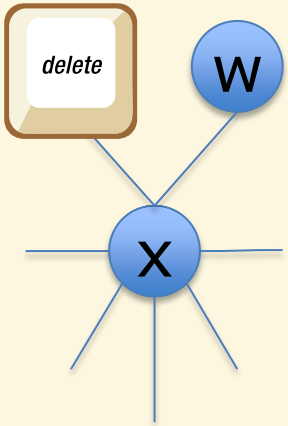


Degree-one vertices



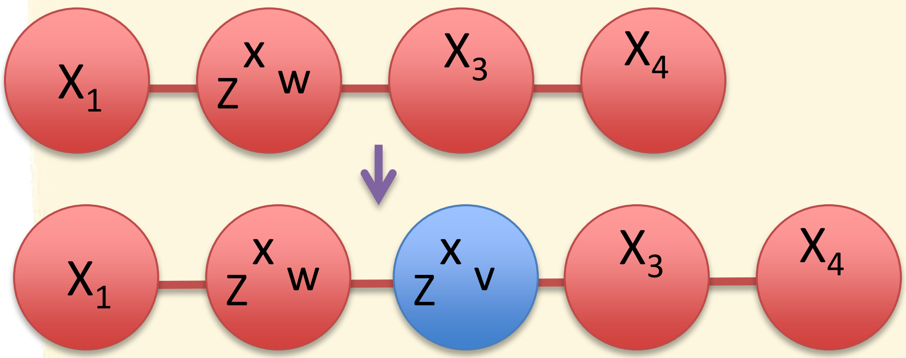
Pathwidth Degree-1 Vertex Rule
If vertex v is only adjacent to x, and
there is another degree-1 vertex w adjacent to x, then
then deleting v is safe for pathwidth k

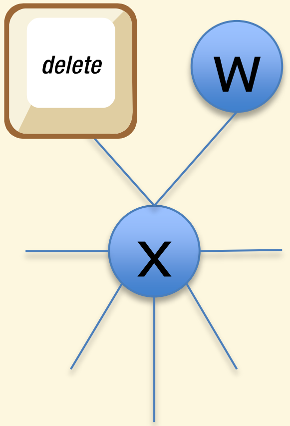
Degree-one vertices



Pathwidth Degree-1 Vertex Rule
If vertex v is only adjacent to x, and
there is another degree-1 vertex w adjacent to x, then
then deleting v is safe for pathwidth k

Degree-one vertices

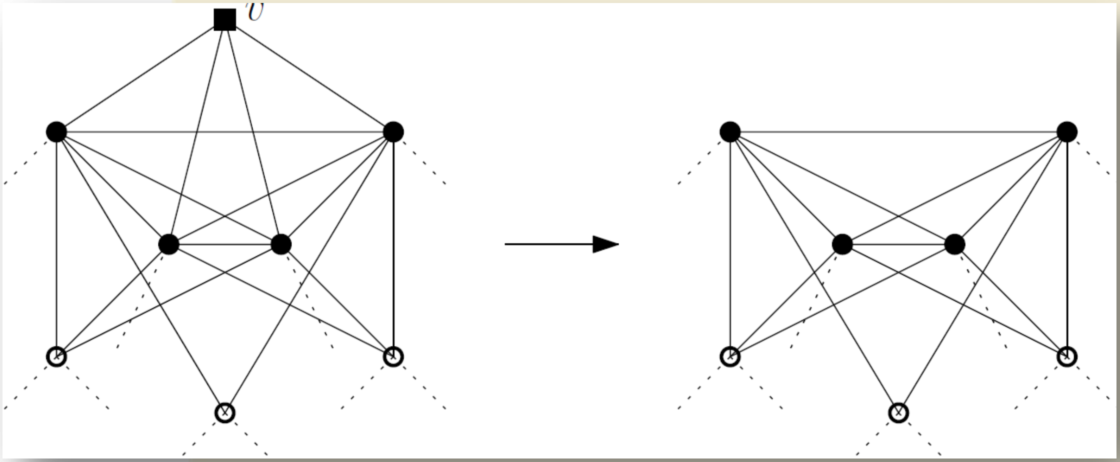




Pathwidth Degree-1 Vertex Rule
If vertex v is only adjacent to x, and
there is another degree-1 vertex w adjacent to x, then
then deleting v is safe for pathwidth k

Pathwidth Simplicial Vertex Rule

Pathwidth Simplicial Vertex Rule



Pathwidth Simplicial Vertex Rule
If v is simplicial with $2 \le \deg(v) \le k$, and
$\forall$ {x,y} in N(v), $\exists$ simplicial vtx $\notin$ N[v] seeing x and y,
then deleting v is safe for pathwidth k

Effects of the reduction rules

- If G has a vertex cover X of size ℓ, and you work relative to the structure of X:
  - Easy counting arguments prove $O(\ell^3)$ vertices
  - (After some trivial rules)

Effects of the reduction rules

- If G has a vertex cover X of size ℓ, and you work relative to the structure of X:
  - Easy counting arguments prove $O(\ell^3)$ vertices
  - (After some trivial rules)

Polynomial kernels (sizes in # vertices)

- $O(\ell^3)$ when ℓ is the vertex cover number
- $O(c\ell^3 + c^2\ell^2)$ when ℓ is the size of a vertex set whose removal gives components of at most c vertices each
- $O(\ell^4)$ when ℓ is the size of a vertex set whose removal results in disjoint stars

A kernelization lower bound

A kernelization lower bound

- Pathwidth of a clique K_t is t-1

A kernelization lower bound

- Pathwidth of a clique K_t is t-1
- Pathwidth is easy for graphs that are “almost” a clique?
  - That become a clique after deleting k vertices

A kernelization lower bound

- Pathwidth of a clique K_t is t-1
- Pathwidth is easy for graphs that are "almost" a clique?
  - That become a clique after deleting k vertices

Pathwidth and Treewidth **do not** admit polynomial kernels parameterized by vertex-deletion distance to a clique (unless NP ⊆ coNP/poly)

A kernelization lower bound

- Pathwidth of a clique K_t is t-1
- Pathwidth is easy for graphs that are "almost" a clique?
  - That become a clique after deleting k vertices
- Builds on the NP-completeness proof for Treewidth and Pathwidth by Arnborg, Corneil & Proskurowski '87
  - They reduce Minimum Cut Linear Arrangement to computing Tree/path width on cobipartite graphs

Pathwidth and Treewidth **do not** admit polynomial kernels parameterized by vertex-deletion distance to a clique (unless NP $\subseteq$ coNP/poly)

A kernelization lower bound

- Pathwidth of a clique K_t is t-1
- Pathwidth is easy for graphs that are "almost" a clique?
  - That become a clique after deleting k vertices
- Builds on the NP-completeness proof for Treewidth and Pathwidth by Arnborg, Corneil & Proskurowski '87
  - They reduce Minimum Cut Linear Arrangement to computing Tree/path width on cobipartite graphs
- We build a cross-composition of MinCut on cubic graphs into tree/pathwidth on a cobipartite graph where one partite set is small

Pathwidth and Treewidth **do not** admit polynomial kernels parameterized by vertex-deletion distance to a clique (unless NP $\subseteq$ coNP/poly)

A kernelization lower bound

- Pathwidth of a clique K_t is t-1
- Pathwidth is easy for graphs that are “almost” a clique?
  - That become a clique after deleting k vertices
- Builds on the NP-completeness proof for Treewidth and Pathwidth by Arnborg, Corneil & Proskurowski ‘87
  - They reduce Minimum Cut Linear Arrangement to computing Tree/path width on cobipartite graphs
- We build a cross-composition of MinCut on cubic graphs into tree/pathwidth on a cobipartite graph where one partite set is small
  - Deleting the small set yields a clique

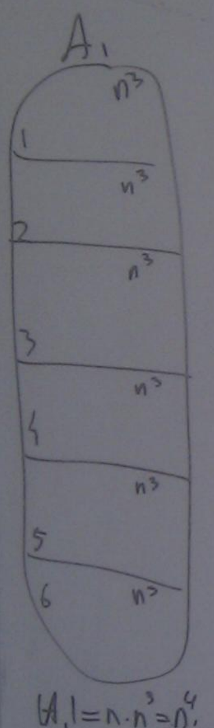
Pathwidth and Treewidth **do not** admit polynomial kernels parameterized by vertex-deletion distance to a clique (unless NP $\subseteq$ coNP/poly)

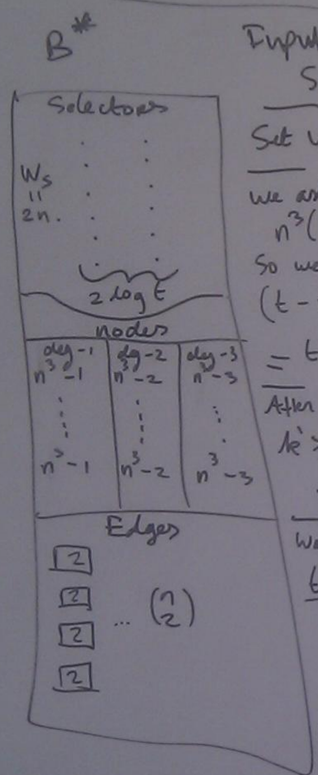
Details of the construction …

Cutwidth crosscomposes into TW by clique deletion.

known: cutwidth is NP-complete on planar max. deg ≤ 3 graphs.

Poly equ.: all input graphs have same degree sequence (characterized by n^4) and ask for same k. Sort vertices by degree. Assume $n \geq 3$.





Input instance on deg ≤ 3 graph has cutwidth $\leq |E| \leq \frac{3n}{2}$.

So $1 \leq k \leq \frac{3n}{2}$.

Set $w_s := 2n$.

we ask for (on small subinstance $(V_i \cup B^*)$):

$n^3(n+1) + k - 1$.

So we ask in the big instance for:

$(t-1)n^4 + n^3(n+1) + k - 1 + 2n \log t$

$= tn^4 + n^3 + k - 1 + 2n \log t$.

After solving instance we can eliminate the rest, provided that

$k' \geq (t-1)n^4 - 1 + 2n(2\log t) + 2\binom{n}{2} + n(n^3-1)$

$= tn^4 - 1 + 4n\log t + 2\binom{n}{2} - n$.

We need:

$\underline{tn^4} + n^3 + k - 1 + 2n\log t \geq \underline{tn^4} - 1 + 4n\log t + 2\binom{n}{2} - n$

$n^3 + k - 1 \geq 2n\log t + 2\binom{n}{2} - n - 1$

$n^3 + k \geq 2n\log t + 2\binom{n}{2} - n$.

Should work if $n \geq \log t$:

Then $2n\log t + 2\binom{n}{2} - n \leq 2n^2 + 2\binom{n}{2} - n \leq 2n^2 + \frac{2n(n-1)}{2} - n$

$\leq 2n^2 + n^2 = 3n^2$. So: $n^3 \geq 3n^2 \iff n \geq 3$.

$n^2 > c(n+1)^k$

$c(n+1)^k \leq c(2n)^k$

$\leq c \cdot 2^k \cdot n^k$.

$n^2 > c \cdot 2^k \cdot n^k$

$\iff n^{2-k} > c \cdot 2^k$

$\iff n = (c \cdot 2^k)^{\frac{1}{2-k}}$.

Zij near tweemacht. $f(n+1) = n^2$.

$f(n) = n$. $f(n+1) = 2^{2^{\lceil \lg \lg (n+1) \rceil}}$

Laat zien: $\lceil \lg \lg (n+1) \rceil = (\lg \lg n) + 1$.

We weten: $\lg \lg (n+1) > \lg \lg n$, en $\lg \lg (n) \% 1 = 0$.

Ons $\lceil \lg \lg (n+1) \rceil \geq \lg \lg n + 1$.

TW b

Conclusion

Conclusion

- Reduction rules for pathwidth are more complicated than for treewidth, because the structure is more restricted

Conclusion

- Reduction rules for pathwidth are more complicated than for treewidth, because the structure is more restricted
- Analysis proves effect of the rules with respect to several parameters

Conclusion

- Reduction rules for pathwidth are more complicated than for treewidth, because the structure is more restricted
- Analysis proves effect of the rules with respect to several parameters
- Pathwidth and treewidth do not admit polynomial kernels by deletion distance to a clique

Future directions

Experimental evaluation of the reduction rules

Lower bounds on kernel sizes

- Kernel with $O(k^{3-\varepsilon})$ bits for parameterization by vertex cover?

Pathwidth parameterized by feedback vertex set

- Polynomial kernel for treewidth by FVS
- Trees have constant treewidth but potentially large pathwidth

Future directions

Experimental evaluation of the reduction rules

Lower boun

- K

THANK YOU!

Pathwidth

- Polynomial kernel for treewidth by FVS
- Trees have constant treewidth but potentially large pathwidth