Recognizing Rank-width Quickly

Sang-il Oum

School of Mathematics
Georgia Institute of Technology

October 18, 2005

Workshop on Graph Classes, Width Parameters and Optimization

Clique-width

a complexity measure of graphs

k-expression: expression on vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$ using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ ($i \neq j$)
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i

$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

Clique-width of G, denoted by $cwd(G)$: minimum k such that G can be expressed by k-expression (after forgetting the labels)

Clique-width
a complexity measure of graphs

k-expression: expression on
vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$
using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ $(i \neq j)$
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i

$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

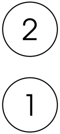
Clique-width of G, denoted by $cwd(G)$:
minimum k such that G can be expressed by k-expression (after forgetting the labels)

Clique-width

a complexity measure of graphs

k-expression: expression on vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$ using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ $(i \neq j)$
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i



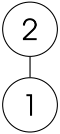
$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

Clique-width of G, denoted by $cwd(G)$: minimum k such that G can be expressed by k-expression (after forgetting the labels)

Clique-width
a complexity measure of graphs

k-expression: expression on
vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$
using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ ($i \neq j$)
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i



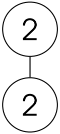
$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

Clique-width of G, denoted by $cwd(G)$:
minimum k such that G can be expressed by k-expression (after forgetting the labels)

Clique-width
a complexity measure of graphs

k-expression: expression on
vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$
using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ $(i \neq j)$
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i



$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

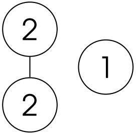
Clique-width of G, denoted by $cwd(G)$:
minimum k such that G can be expressed by k-expression (after forgetting the labels)

Clique-width

a complexity measure of graphs

k-expression: expression on vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$ using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ $(i \neq j)$
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i

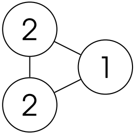


$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

Clique-width of G, denoted by $cwd(G)$:
minimum k such that G can be expressed by k-expression (after forgetting the labels)

k-expression: expression on
vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$
using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ $(i \neq j)$
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i



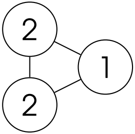
$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

Clique-width of G, denoted by $cwd(G)$:
minimum k such that G can be expressed by k-expression (after forgetting the labels)

Clique-width
a complexity measure of graphs

k-expression: expression on
vertex-labeled graphs with labels $\{1, 2, \cdots, k\}$
using the following 4 operations

$G_1 \oplus G_2$	disjoint union of G_1 and G_2
$\eta_{i,j}(G)$	add edges uv s.t. $lab(u) = i$, $lab(v) = j$ $(i \neq j)$
$\rho_{i \to j}(G)$	relabel all vertices of label i into label j
$\cdot_i$	create a graph with one vertex with label i



$$\eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\cdot_2 \oplus \cdot_1)) \oplus \cdot_1)$$

Clique-width of G, denoted by $cwd(G)$:
minimum k such that G can be expressed by k-expression (after forgetting the labels)

Small clique-width is good for algorithms. Many NP-hard problems are solvable in poly time for graphs of small clique-width by dynamic programming techniques with the k-expressions.

Problems

- Construction: How to find a k-expression quickly if there is one? (k:fixed)
- Decision: How to decide that clique-width $\leq k$? (k:fixed)

- Difficult to tell that clique-width is large.
  (Is the decision problem in coNP?)
- Instead, we study *rank-width*.
  Small rank-width $\Leftrightarrow$ Small clique-width

$$\text{Rank-width} \leq \text{Clique-width} \leq 2^{1+\text{Rank-width}} - 1.$$

Small clique-width is good for algorithms. Many NP-hard problems are solvable in poly time for graphs of small clique-width by dynamic programming techniques with the k-expressions.

Problems

- Construction: How to find a k-expression quickly if there is one? (k:fixed)
- Decision: How to decide that clique-width $\leq k$? (k:fixed)

- Difficult to tell that clique-width is large.
  (Is the decision problem in coNP?)
- Instead, we study *rank-width*.
  Small rank-width $\Leftrightarrow$ Small clique-width

$$\text{Rank-width} \leq \text{Clique-width} \leq 2^{1+\text{Rank-width}} - 1.$$

How to decide that rank-width $\leq k$ quickly?

Outline:

- definition of rank-width,
- reduction to bipartite graphs.

Previous "slower" algorithm:

- $O(n^9 \log n)$

  Combining 3 papers { Seymour, Oum (2002), Oum (2004), Courcelle, Oum (2004).

- Later improved to $O(n^4)$.

How to decide that rank-width $\leq k$ quickly?

Outline:

- definition of rank-width,
- reduction to bipartite graphs.

Previous "slower" algorithm:

- $O(n^9 \log n)$

  Combining 3 papers { Seymour, Oum (2002), Oum (2004), Courcelle, Oum (2004).

- Later improved to $O(n^4)$.

How to decide that rank-width $\leq k$ quickly?

Outline:

- definition of rank-width,
- reduction to bipartite graphs.

Previous "slower" algorithm:

- $O(n^9 \log n)$

  Combining 3 papers $\begin{cases} \text{Seymour, Oum (2002),} \\ \text{Oum (2004),} \\ \text{Courcelle, Oum (2004).} \end{cases}$

- Later improved to $O(n^4)$.

Branch-width of Symmetric Submodular Functions

$f : 2^V \to \mathbb{Z}$ is

- symmetric if $f(X) = f(V \setminus X)$ for all $X \subseteq V$,
- submodular if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq V$.

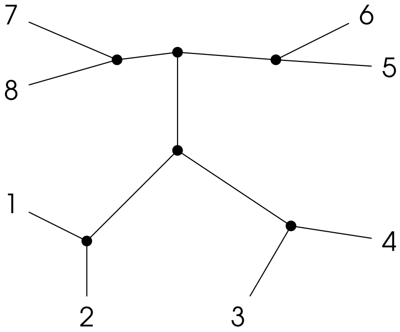
Branch-decomposition of f: a pair (T, L) of a subcubic tree T and a bijection $L : V \to$ leaves of T}.

Branch-width of Symmetric Submodular Functions

$f : 2^V \to \mathbb{Z}$ is

- symmetric if $f(X) = f(V \setminus X)$ for all $X \subseteq V$,
- submodular if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq V$.

Branch-decomposition of f: a pair (T, L) of a subcubic tree T and a bijection $L : V \to$ leaves of T}.

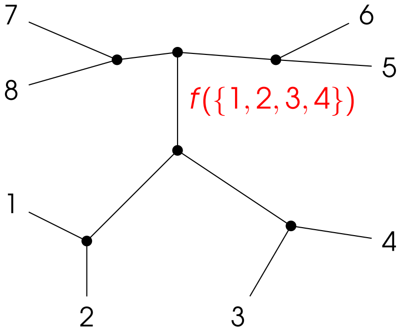


Branch-width of Symmetric Submodular Functions

$f : 2^V \to \mathbb{Z}$ is

- symmetric if $f(X) = f(V \setminus X)$ for all $X \subseteq V$,
- submodular if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq V$.

Branch-decomposition of f: a pair (T, L) of a subcubic tree T and a bijection $L : V \to$ leaves of T}.

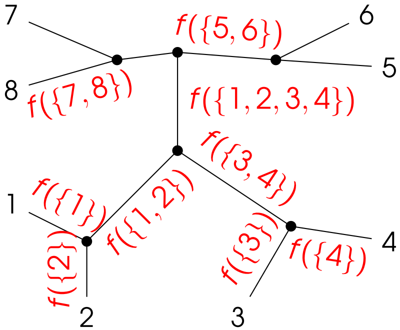


Width of an edge e of T: $f(A_e)$
(A_e, B_e) is a partition of V given by deleting e.

$f : 2^V \to \mathbb{Z}$ is

- symmetric if $f(X) = f(V \setminus X)$ for all $X \subseteq V$,
- submodular if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq V$.

Branch-decomposition of f: a pair (T, L) of a subcubic tree T and a bijection $L : V \to$ leaves of T}.



Width of an edge e of T: $f(A_e)$
(A_e, B_e) is a partition of V given by deleting e.

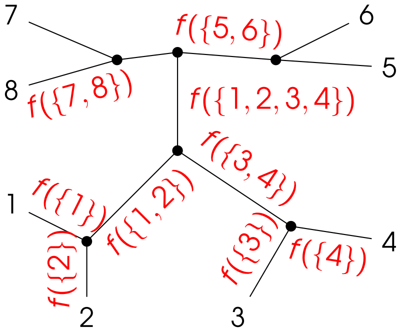
Width of (T, L): $\max_e \text{width}(e)$

Branch-width of Symmetric Submodular Functions

$f : 2^V \to \mathbb{Z}$ is

- symmetric if $f(X) = f(V \setminus X)$ for all $X \subseteq V$,
- submodular if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq V$.

Branch-decomposition of f: a pair (T, L) of a subcubic tree T and a bijection $L : V \to$ leaves of T}.



Width of an edge e of T: $f(A_e)$
(A_e, B_e) is a partition of V given by deleting e.

Width of (T, L): $\max_e \text{width}(e)$

Rank-width: $min_{(T,L)} \text{width}(T, L)$.

- Rank-width of a graph:
  Branch-width of the cut-rank function ρ of the graph.

  $\rho_G(X) = \text{rank}(\text{submatrix of } M \text{ with rows } X, \text{ columns } V(G) \setminus X).$

  where M is the adjacency matrix over $\mathrm{GF}(2)$.
- Branch-width of a matroid:
  Branch-width of the connectivity function of the matroid

$$\lambda_{\mathcal{M}}(X) = r(X) + r(E(\mathcal{M}) \setminus X) - r(E(\mathcal{M})) + 1.$$

- cf: Carving-width of a graph:
  Branch-width of $e(X)$ where

$$e(X) = \text{number of edges meeting both } X \text{ and } V \setminus X.$$

Easy for Bipartite Graphs

- Branch-width(Binary Matroid $\mathcal{M}$)
  = Rank-width(Fundamental Graph of $\mathcal{M}$) + 1.
- Hliněný (2002):
  For fixed k, $O(n^3)$-time algorithm to decide whether

  $$\text{Branch-width (Binary Matroid)} \leq k + 1.$$

  $n = |E(\mathcal{M})|$ and the input matroid is given by matrix representation.

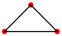
For fixed k, there is a $O(n^3)$-time algorithm to decide

$$\text{Rank-width (Bipartite Graph)} \leq k.$$

Reduction to Bipartite Graphs

Graph $G = (V, E) \Longrightarrow$ Bipartite graph $B(G)$ Courcelle (2004)

- $(v, 1), (v, 2), (v, 3), (v, 4)$ are vertices of $B(G)$ corresponding to $v \in V$.
- $(v, 1)$ is adjacent to $(w, 4)$ in $B(G)$ iff $vw \in E$.
- $(v, 1)(v, 2)(v, 3)(v, 4)$ is a 3-edge path for each v.



G

$B(G)$

Theorem

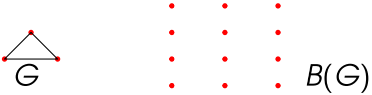
If $E(G) \neq \emptyset$, then Rank-width(B(G)) = 2 Rank-width(G).

How to prove this?

Reduction to Bipartite Graphs

Graph $G = (V, E) \Longrightarrow$ Bipartite graph $B(G)$ Courcelle (2004)

- $(v, 1), (v, 2), (v, 3), (v, 4)$ are vertices of $B(G)$ corresponding to $v \in V$.
- $(v, 1)$ is adjacent to $(w, 4)$ in $B(G)$ iff $vw \in E$.
- $(v, 1)(v, 2)(v, 3)(v, 4)$ is a 3-edge path for each v.



Theorem

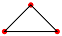
If $E(G) \neq \emptyset$, then Rank-width(B(G)) = 2 Rank-width(G).

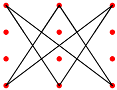
How to prove this?

Reduction to Bipartite Graphs

Graph $G = (V, E) \Longrightarrow$ Bipartite graph $B(G)$ Courcelle (2004)

- $(v, 1), (v, 2), (v, 3), (v, 4)$ are vertices of $B(G)$ corresponding to $v \in V$.
- $(v, 1)$ is adjacent to $(w, 4)$ in $B(G)$ iff $vw \in E$.
- $(v, 1)(v, 2)(v, 3)(v, 4)$ is a 3-edge path for each v.





Theorem

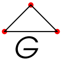
If $E(G) \neq \emptyset$, then Rank-width(B(G)) = 2 Rank-width(G).

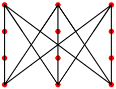
How to prove this?

Reduction to Bipartite Graphs

Graph $G = (V, E) \Longrightarrow$ Bipartite graph $B(G)$ Courcelle (2004)

- $(v, 1), (v, 2), (v, 3), (v, 4)$ are vertices of $B(G)$ corresponding to $v \in V$.
- $(v, 1)$ is adjacent to $(w, 4)$ in $B(G)$ iff $vw \in E$.
- $(v, 1)(v, 2)(v, 3)(v, 4)$ is a 3-edge path for each v.





Theorem

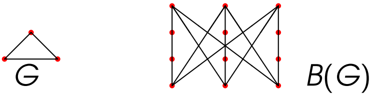
If $E(G) \neq \emptyset$, then Rank-width(B(G)) = 2 Rank-width(G).

How to prove this?

Reduction to Bipartite Graphs

Graph $G = (V, E) \Longrightarrow$ Bipartite graph $B(G)$ Courcelle (2004)

- $(v, 1), (v, 2), (v, 3), (v, 4)$ are vertices of $B(G)$ corresponding to $v \in V$.
- $(v, 1)$ is adjacent to $(w, 4)$ in $B(G)$ iff $vw \in E$.
- $(v, 1)(v, 2)(v, 3)(v, 4)$ is a 3-edge path for each v.



Theorem

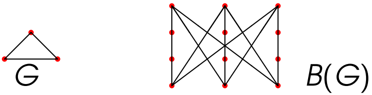
If $E(G) \neq \emptyset$, then Rank-width(B(G)) = 2 Rank-width(G).

How to prove this?

Reduction to Bipartite Graphs

Graph $G = (V, E) \Longrightarrow$ Bipartite graph $B(G)$ Courcelle (2004)

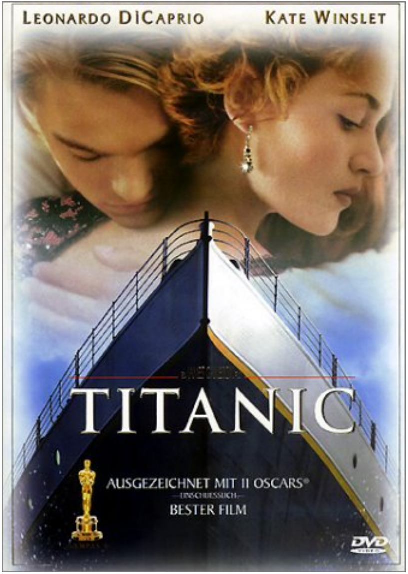
- $(v, 1)$, $(v, 2)$, $(v, 3)$, $(v, 4)$ are vertices of $B(G)$ corresponding to $v \in V$.
- $(v, 1)$ is adjacent to $(w, 4)$ in $B(G)$ iff $vw \in E$.
- $(v, 1)(v, 2)(v, 3)(v, 4)$ is a 3-edge path for each v.



Theorem

If $E(G) \neq \emptyset$, then Rank-width(B(G)) $= 2$ Rank-width(G).

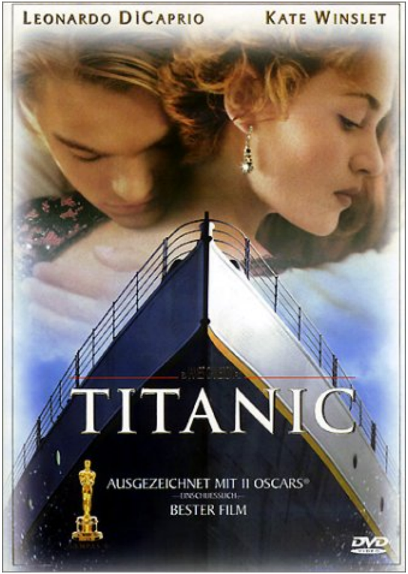
How to prove this?



We need the notion of titanic set. (Robertson and Seymour)

X is titanic if

for all 3-partition A, B, C of X, $\max(\rho_G(A), \rho_G(B), \rho_G(C)) \geq \rho_G(X)$.

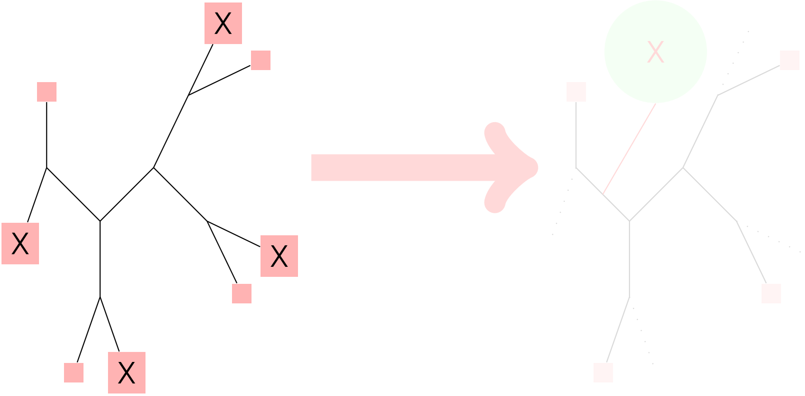


We need the notion of titanic set.
(Robertson and Seymour)

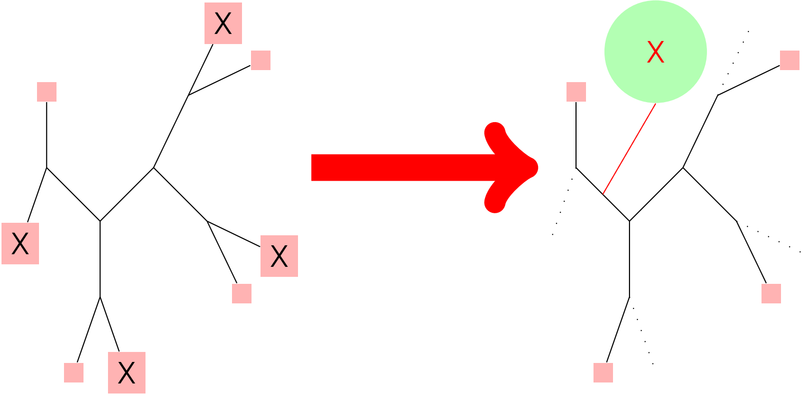
X is titanic if

for all 3-partition A, B, C of X,
$\max(\rho_G(A), \rho_G(B), \rho_G(C)) \geq \rho_G(X)$.

If (T, L) is a rank-decomposition of width k and X is titanic, then one can arrange X together so that every edge out of X in T has width $\leq k$.
(Robertson and Seymour)



If (T, L) is a rank-decomposition of width k and X is titanic, then one can arrange X together so that every edge out of X in T has width $\leq k$.
(Robertson and Seymour)



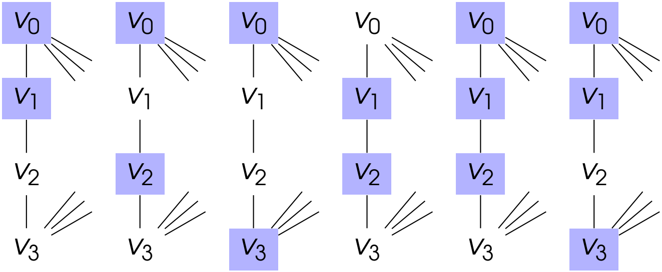
Every 3-edge Vertical Path is Titanic

Want to show: Every 3-partition A, B, C of $\{v_0, v_1, v_2, v_3\}$ satisfies

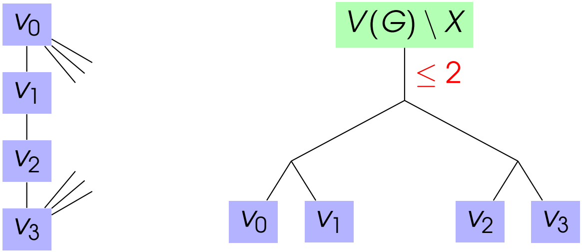
$$\max(\rho(A), \rho(B), \rho(C)) \geq \rho(X).$$

WMA: $\rho(X) = 2$.

Enough to show: If $A \subset X$ and $2 \leq |A| \leq 3$, then $\rho(A) \geq 2$.



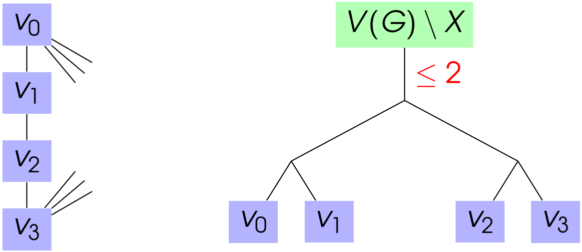
We can arrange copies of a vertex into a rooted binary tree so that every edge have width at most 2.



We obtain a branch-decomposition (T, L) of $B(G)$ such that each vertical 3-edge path occurs as a separation.

Then we transform (T, L) of width $2k$ into a branch-decomposition (T', L') of G of width k simply by grouping those 4 vertices into a leaf.

We can arrange copies of a vertex into a rooted binary tree so that every edge have width at most 2.



We obtain a branch-decomposition (T, L) of $B(G)$ such that each vertical 3-edge path occurs as a separation.

Then we transform (T, L) of width $2k$
into a branch-decomposition (T', L') of G of width k
simply by grouping those 4 vertices into a leaf.

Discussion: Construction Problem

- (OPEN) Is there a poly-time algorithm to construct a rank-decomposition of width $\leq k$ if rank-width $\leq k$?
- (SOLVED) There is a poly-time algorithm to construct a rank-decomposition of width $\leq f(k)$ if rank-width $\leq k$.
  - $f(k) = 3k + 1$, $O(n^9 \log n)$. Seymour, Oum.
  - $f(k) = 3k + 1$, $O(n^4)$ Oum.
  - $f(k) = 12k$, $O(n^3)$ Oum.
    By reducing to binary matroids and then use Hliněný's algorithm.

Problem: Hliněný's algorithm does not output a 'clean' rank-decomposition of $B(G)$: we have to work on the output to get the rank-decomposition of G.

Hliněný emailed me (July 31) claiming that it is possible to modify his previous algorithm to output a clean rank-decomposition. This would mean

$$f(k) = 3k.$$

Discussion: Construction Problem

- (OPEN) Is there a poly-time algorithm to construct a rank-decomposition of width $\leq k$ if rank-width $\leq k$?
- (SOLVED) There is a poly-time algorithm to construct a rank-decomposition of width $\leq f(k)$ if rank-width $\leq k$.
  - $f(k) = 3k + 1$, $O(n^9 \log n)$. Seymour, Oum.
  - $f(k) = 3k + 1$, $O(n^4)$ Oum.
  - $f(k) = 12k$, $O(n^3)$ Oum.
    By reducing to binary matroids and then use Hliněný's algorithm.

Problem: Hliněný's algorithm does not output a 'clean' rank-decomposition of $B(G)$: we have to work on the output to get the rank-decomposition of G.

Hliněný emailed me (July 31) claiming that it is possible to modify his previous algorithm to output a clean rank-decomposition. This would mean

$$f(k) = 3k.$$

Discussion: Construction Problem

- (OPEN) Is there a poly-time algorithm to construct a rank-decomposition of width $\leq k$ if rank-width $\leq k$?
- (SOLVED) There is a poly-time algorithm to construct a rank-decomposition of width $\leq f(k)$ if rank-width $\leq k$.
  - $f(k) = 3k + 1$, $O(n^9 \log n)$. Seymour, Oum.
  - $f(k) = 3k + 1$, $O(n^4)$ Oum.
  - $f(k) = 12k$, $O(n^3)$ Oum.
    By reducing to binary matroids and then use Hliněný's algorithm.

Problem: Hliněný's algorithm does not output a `clean' rank-decomposition of $B(G)$: we have to work on the output to get the rank-decomposition of G.

Hliněný emailed me (July 31) claiming that it is possible to modify his previous algorithm to output a clean rank-decomposition. This would mean

$$f(k) = 3k.$$

Discussion: Construction Problem

- (OPEN) Is there a poly-time algorithm to construct a rank-decomposition of width $\leq k$ if rank-width $\leq k$?
- (SOLVED) There is a poly-time algorithm to construct a rank-decomposition of width $\leq f(k)$ if rank-width $\leq k$.
  - $f(k) = 3k + 1$, $O(n^9 \log n)$. Seymour, Oum.
  - $f(k) = 3k + 1$, $O(n^4)$ Oum.
  - $f(k) = 12k$, $O(n^3)$ Oum.
    By reducing to binary matroids and then use Hliněný's algorithm.

Problem: Hliněný's algorithm does not output a 'clean' rank-decomposition of $B(G)$: we have to work on the output to get the rank-decomposition of G.

Hliněný emailed me (July 31) claiming that it is possible to modify his previous algorithm to output a clean rank-decomposition. This would mean

$$f(k) = 3k.$$