Finding Branch-decompositions & Rank-decompositions

Sang-il Oum

Dept. of Mathematical Sciences
KAIST (Korea Adv. Institute of Sci. and Tech.)
Daejeon, Korea.

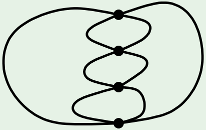
April 7, 2008

Joint work with Petr Hliněný

Workshop on Graph Decompositions:
Theoretical, Algorithmic and Logical Aspects
CIRM, Luminy, Marseille (France)

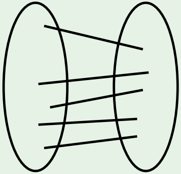
Connectivity

Partition (E, F) of $E(G)$:



$v(X)$ =#vertices meeting both X and $E \setminus X$.

Partition (E, F) of $V(G)$:



$e(X)$ =#edges meeting both X and $V \setminus X$.

M: matroid, $\lambda(X) = r(X) + r(E(M) - X) - r(E(M))$.

A function $f : 2^V \to \mathbb{Z}$ is a connectivity function if

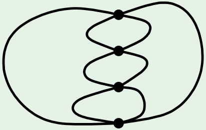
(i) $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$, (submodular)

(ii) $f(X) = f(V \setminus X)$, (symmetric)

(iii) $f(\emptyset) = 0$.

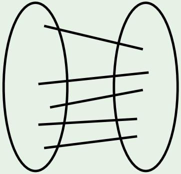
Connectivity

Partition (E, F) of $E(G)$:



$v(X)$ =#vertices meeting both X and $E \setminus X$.

Partition (E, F) of $V(G)$:



$e(X)$ =#edges meeting both X and $V \setminus X$.

M: matroid, $\lambda(X) = r(X) + r(E(M) - X) - r(E(M))$.

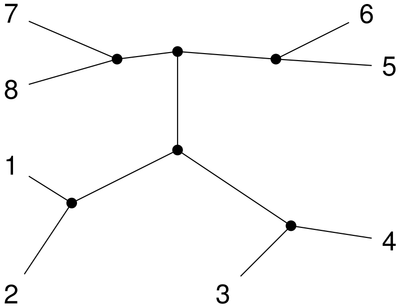
A function $f : 2^V \to \mathbb{Z}$ is a connectivity function if

(i) $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$, (submodular)

(ii) $f(X) = f(V \setminus X)$, (symmetric)

(iii) $f(\emptyset) = 0$.

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



Branch-width

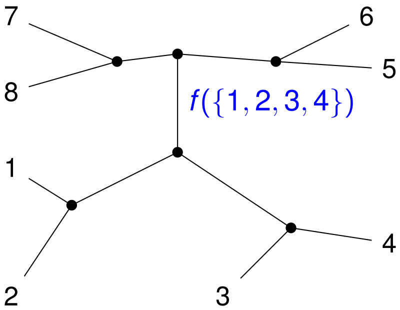
$V = E(G)$

Carving-width

$V = V(G)$

Branch-width of matroids
(Branch-width of λ) + 1.
$\lambda(X) =$
$r(X) + r(E(M) - X) - r(E(M))$.
$V = E(M)$.

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



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

Branch-width

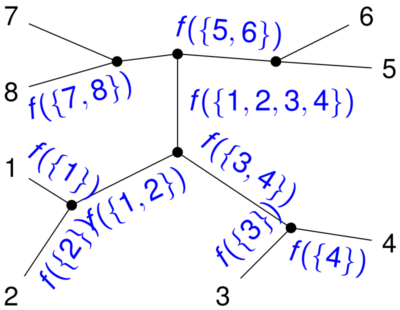
$V = E(G)$

Carving-width

$V = V(G)$

Branch-width of matroids
(Branch-width of λ) + 1.
$\lambda(X) =$
$r(X) + r(E(M) - X) - r(E(M))$.
$V = E(M)$.

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



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

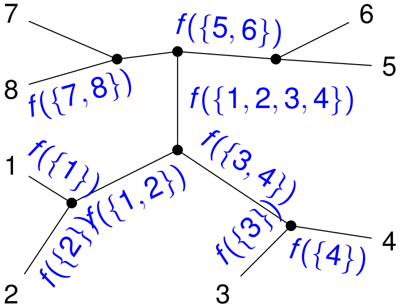
$V = E(G)$

Carving-width

$V = V(G)$

Branch-width of matroids
(Branch-width of λ) $+ 1$.
$\lambda(X) =$
$r(X) + r(E(M) - X) - r(E(M))$.
$V = E(M)$.

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



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: $min_{(T,L)}$ width(T, L).
(If $|V| \leq 1$, then branch-width=0)

Branch-width

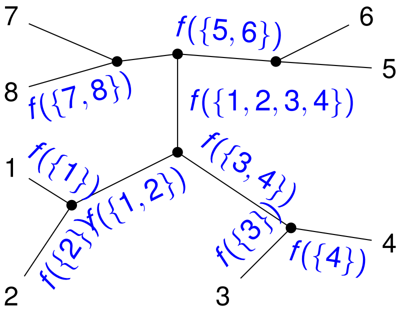
$V = E(G)$

Carving-width

$V = V(G)$

Branch-width of matroids
(Branch-width of λ) $+ 1$.
$\lambda(X) =$
$r(X) + r(E(M) - X) - r(E(M))$.
$V = E(M)$.

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

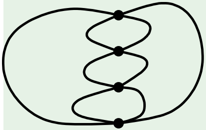


Width of an edge e of T: $f(A_e)$
(A_e, B_e) is a partition of V given by deleting e.

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

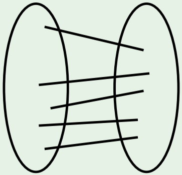
Branch-width: $min_{(T,L)}$ width(T, L).
(If $|V| \leq 1$, then branch-width=0)

Branch-width



$V = E(G)$

Carving-width



$V = V(G)$

Branch-width of matroids
(Branch-width of λ) + 1.
$\lambda(X) =$
$r(X) + r(E(M) - X) - r(E(M))$.
$V = E(M)$.

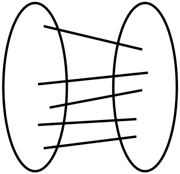
Branch-width is “good”

Deciding whether Branch-width $\leq k$ for fixed k

- Branch-width of graphs: Linear (Bodlaender, Thilikos ’97)
- Carving-width of graphs: Linear (Thilikos, Serna, Bodlaendar ’00)
- Branch-width of matroids represented over a fixed finite field: $O(|E(M)|^3)$ (Hliněný ’05)
- Any connectivity function: $O(\gamma n^{8k+6} \log n)$ (O., Seymour ’07)

Cut-rank function: another connectivity function

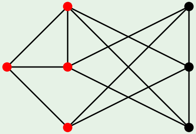
$(X \quad , \quad Y)$: partition of $V(G)$



$$\rho_G(X) = \mathrm{rank} \begin{array}{cc} & Y \\ X & \begin{pmatrix} \text{0-1 matrix} \end{pmatrix} \end{array}$$

(The matrix is over the binary field $\mathrm{GF}(2)$.)

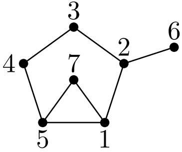
$$\rho(\text{red vertices}) = \mathrm{rank} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = 2.$$



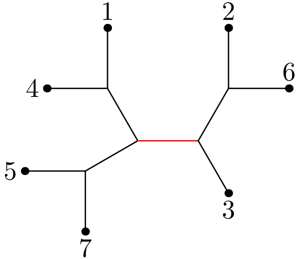
Rank-width

Definition of Rank-width

Rank-width of a graph G = Branch-width of the cut-rank function ρ_G



Graph



Rank-decomposition
Width= 2

Rank-width: min width(rank-decomposition).

Clique-width

Courcelle, Engelfriet, and Rozenberg '93 / Courcelle, Olariu '00

- *k*-expression: algebraic expression on vertex-labelled graphs with k labels $1, 2, \ldots, k$.
  - $\cdot_i$ a single vertex with label i
  - $G_1 \oplus G_2$ disjoint union
  - $\rho_{i \to j}(G)$ relabel vertices of label i into j
  - $\eta_{i,j}(G)$ $(i \neq j)$ add edges vertices of label i and j
- Clique-width of a graph G:
  min k such that G has a k-expression.

$G_1 = \eta_{1,2}(\cdot_1 \oplus \cdot_2)$

$G_2 = \rho_{2 \to 1}(G_1) \oplus \cdot_2$

$G_3 = \eta_{1,2}(G_2)$

Rank-width and clique-width are 'equivalent' (O., Seymour '06)

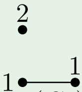
$$\mathrm{rwd}(G) \leq \mathrm{cwd}(G) \leq 2^{\mathrm{rwd}(G)+1} - 1.$$

Clique-width

Courcelle, Engelfriet, and Rozenberg ’93 / Courcelle, Olariu ’00

- *k*-expression: algebraic expression on vertex-labelled graphs with k labels $1, 2, \ldots, k$.
  - $\cdot_i$ a single vertex with label i
  - $G_1 \oplus G_2$ disjoint union
  - $\rho_{i \to j}(G)$ relabel vertices of label i into j
  - $\eta_{i,j}(G)$ $(i \neq j)$ add edges vertices of label i and j
- Clique-width of a graph G:
  min k such that G has a k-expression.

$G_1 = \eta_{1,2}(\cdot_1 \oplus \cdot_2)$



$G_2 = \rho_{2\to 1}(G_1) \oplus \cdot_2$

$G_3 = \eta_{1,2}(G_2)$

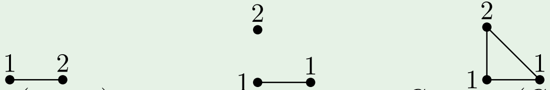
Rank-width and clique-width are ‘equivalent’ (O., Seymour ’06)

$$\mathrm{rwd}(G) \leq \mathrm{cwd}(G) \leq 2^{\mathrm{rwd}(G)+1} - 1.$$

Clique-width

Courcelle, Engelfriet, and Rozenberg '93 / Courcelle, Olariu '00

- *k*-expression: algebraic expression on vertex-labelled graphs with k labels $1, 2, \ldots, k$.
  - $\cdot_i$ a single vertex with label i
  - $G_1 \oplus G_2$ disjoint union
  - $\rho_{i \to j}(G)$ relabel vertices of label i into j
  - $\eta_{i,j}(G)$ $(i \neq j)$ add edges vertices of label i and j
- Clique-width of a graph G:
  min k such that G has a k-expression.



$G_1 = \eta_{1,2}(\cdot_1 \oplus \cdot_2)$ $\quad G_2 = \rho_{2\to 1}(G_1) \oplus \cdot_2$ $\quad G_3 = \eta_{1,2}(G_2)$

Rank-width and clique-width are 'equivalent' (O., Seymour '06)

$$\mathrm{rwd}(G) \leq \mathrm{cwd}(G) \leq 2^{\mathrm{rwd}(G)+1} - 1.$$

Solvable problems when rank-width is bounded (I)

Courcelle, Makowsky, and Rotics '00

Every graph problem expressible in *monadic second-order logic formula* (with no edge-set variables) is solvable in time $O(n^3)$ for graphs having rank-width at most k for fixed k.

CMR'00: Minimize $w(X)$ satisfying $\varphi(X)$ for graphs of bounded rank-width.
CMR'01: Counting the number of true assignments in polynomial time. (assuming unit time for arithmetic operations on $\mathbb{R}$.)

Can I find a partition of vertices into three subsets such that each set has no edges inside? (graph 3-coloring problem)

$$\begin{aligned}\exists X_1 \exists X_2 \exists X_3 \forall v \forall w (v, w \in X_1 \Rightarrow \neg\, \mathrm{adj}(v,w))\\ \wedge \forall v \forall w (v, w \in X_2 \Rightarrow \neg\, \mathrm{adj}(v,w))\\ \wedge \forall v \forall w (v, w \in X_3 \Rightarrow \neg\, \mathrm{adj}(v,w)) \cdots\end{aligned}$$

Solvable problems when rank-width is bounded (II)

Many other problems (that are not MS_1 expressible) can be also solved in polynomial time for graphs of bounded rank-width.

- Finding a chromatic number. (Kobler and Rotics '03)
- Deciding whether a graph has a Hamiltonian cycle. (Wanke '94)
- Given a monadic second-order logic formula φ, list all m such that there is a partition $(X_1, \ldots, X_m)$ of $V(G)$ such that $\varphi(X_i)$ is satisfied for all i. (Rao '07)

All of these algorithms

- need the rank-decomposition of width $\leq k$ as an input, and
- use the dynamic programming.

Solvable problems when rank-width is bounded (II)

Many other problems (that are not MS_1 expressible) can be also solved in polynomial time for graphs of bounded rank-width.

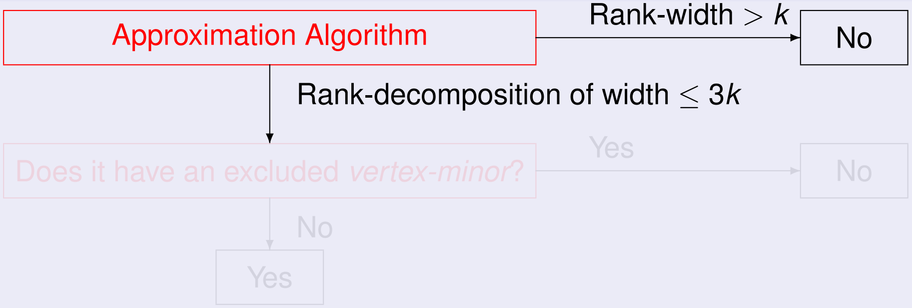
- Finding a chromatic number. (Kobler and Rotics ’03)
- Deciding whether a graph has a Hamiltonian cycle. (Wanke ’94)
- Given a monadic second-order logic formula φ, list all m such that there is a partition $(X_1, \ldots, X_m)$ of $V(G)$ such that $\varphi(X_i)$ is satisfied for all i. (Rao ’07)

All of these algorithms

- need the rank-decomposition of width $\leq k$ as an input, and
- use the dynamic programming.

Previous decision algorithm for rank-width

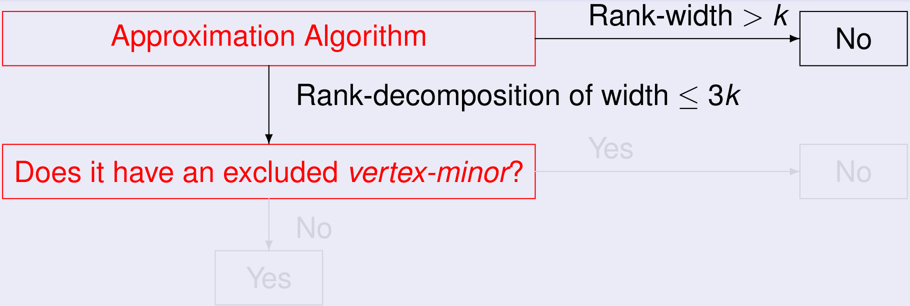
Is rank-width $\leq k$?



- For each k, there are finitely many excluded vertex-minors for the set of graphs of rank-width $\leq k$.
- For a fixed graph H, there is a modulo-2 counting monadic second-order logic formula φ_H to test whether H is a vertex-minor of G.
- It does NOT output the rank-decomposition of width $\leq k$ for Yes instances.

Previous decision algorithm for rank-width

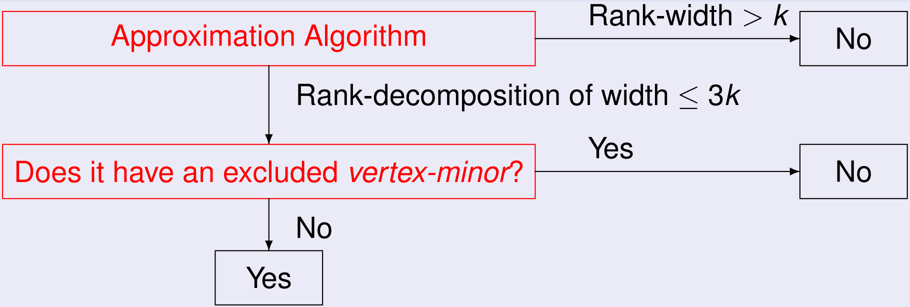
Is rank-width $\leq k$?



- For each k, there are finitely many excluded vertex-minors for the set of graphs of rank-width $\leq k$.
- For a fixed graph H, there is a modulo-2 counting monadic second-order logic formula φ_H to test whether H is a vertex-minor of G.
- It does NOT output the rank-decomposition of width $\leq k$ for Yes instances.

Previous decision algorithm for rank-width

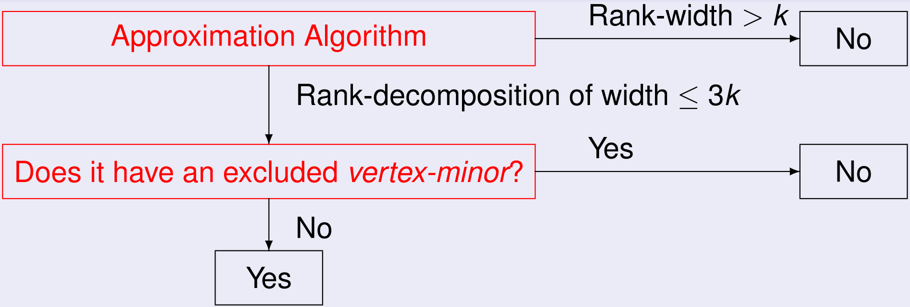
Is rank-width $\leq k$?



- For each k, there are finitely many excluded vertex-minors for the set of graphs of rank-width $\leq k$.
- For a fixed graph H, there is a modulo-2 counting monadic second-order logic formula φ_H to test whether H is a vertex-minor of G.
- It does NOT output the rank-decomposition of width $\leq k$ for Yes instances.

Previous decision algorithm for rank-width

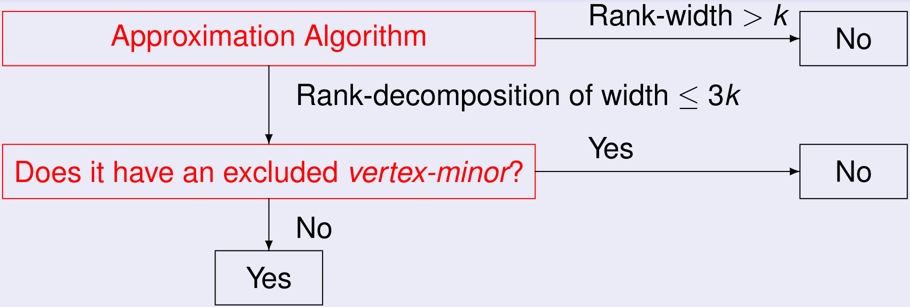
Is rank-width $\leq k$?



- For each k, there are finitely many excluded vertex-minors for the set of graphs of rank-width $\leq k$.
- For a fixed graph H, there is a modulo-2 counting monadic second-order logic formula φ_H to test whether H is a vertex-minor of G.
- It does NOT output the rank-decomposition of width $\leq k$ for Yes instances.

Previous decision algorithm for rank-width

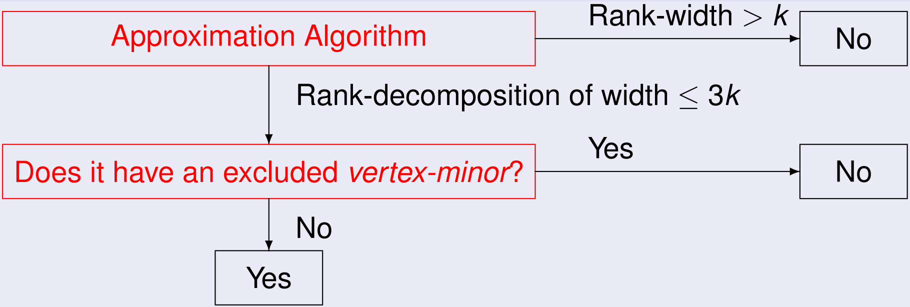
Is rank-width $\leq k$?



- For each k, there are finitely many excluded vertex-minors for the set of graphs of rank-width $\leq k$.
- For a fixed graph H, there is a modulo-2 counting monadic second-order logic formula φ_H to test whether H is a vertex-minor of G.
- It does NOT output the rank-decomposition of width $\leq k$ for Yes instances.

Previous decision algorithm for rank-width

Is rank-width $\leq k$?



- For each k, there are finitely many excluded vertex-minors for the set of graphs of rank-width $\leq k$.
- For a fixed graph H, there is a modulo-2 counting monadic second-order logic formula φ_H to test whether H is a vertex-minor of G.
- It does NOT output the rank-decomposition of width $\leq k$ for Yes instances.

Previous decision algorithm for branch-width

Deciding branch-width $\leq k$

Any connectivity function: $O(\gamma n^{8k+6} \log n)$ (O., Seymour '07)

Suppose that branch-width $\leq k$ (for a connectivity function).

How can we construct a branch-decomposition of width $\leq k$?

Jim Geelen (2005, in O., Seymour '07)

- We can test branch-width of connectivity functions induced by partitions of V (by treating each part as one element).
- Recursively find a pair $a, b \in V$ such that merging them does not increase branch-width. Merge them in one part.

We can construct, in time $O(\gamma n^{8k+9} \log n)$,

- rank-decomposition of width $\leq k$ (if rwd $\leq k$)
- branch-decomposition of width$\leq k$ (if bwd $\leq k$) for matroids.

Previous decision algorithm for branch-width

Deciding branch-width$\leq k$

Any connectivity function: $O(\gamma n^{8k+6} \log n)$ (O., Seymour '07)

Suppose that branch-width $\leq k$ (for a connectivity function).

How can we construct a branch-decomposition of width $\leq k$?

Jim Geelen (2005, in O., Seymour '07)

- We can test branch-width of connectivity functions induced by partitions of V (by treating each part as one element).
- Recursively find a pair $a, b \in V$ such that merging them does not increase branch-width. Merge them in one part.

We can construct, in time $O(\gamma n^{8k+9} \log n)$,

- rank-decomposition of width $\leq k$ (if rwd $\leq k$)
- branch-decomposition of width$\leq k$ (if bwd $\leq k$) for matroids.

We present:

Fixed-parameter-tractable algorithm to construct

- rank-decomposition of width $\leq k$ (if rwd $\leq k$)
- branch-decomposition of width$\leq k$ (if bwd $\leq k$)
  for matroids represented over a fixed finite field.

An essential step is:

Can we test branch-width of a partitioned matroid $\leq k$?

- Partition= disjoint nonempty subsets of V whose union is V.
- Partitioned matroid:
  a matroid with a partition of the element set.
- Branch-width of a partitioned matroid:
  treat each part as a single element.

Then recursively find a pair a, b such that merging them does not increase branch-width. Merge them in one part and repeat.

We present:

Fixed-parameter-tractable algorithm to construct

- rank-decomposition of width $\leq k$ (if rwd $\leq k$)
- branch-decomposition of width$\leq k$ (if bwd $\leq k$)

  for matroids represented over a fixed finite field.

An essential step is:

Can we test branch-width of a partitioned matroid $\leq k$?

- Partition= disjoint nonempty subsets of V whose union is V.
- Partitioned matroid:
  a matroid with a partition of the element set.
- Branch-width of a partitioned matroid:
  treat each part as a single element.

Then recursively find a pair a, b such that merging them does not increase branch-width. Merge them in one part and repeat.

We present:

Fixed-parameter-tractable algorithm to construct

- rank-decomposition of width $\leq k$ (if rwd $\leq k$)
- branch-decomposition of width $\leq k$ (if bwd $\leq k$)

  for matroids represented over a fixed finite field.

An essential step is:

Can we test branch-width of a partitioned matroid $\leq k$?

- Partition= disjoint nonempty subsets of V whose union is V.
- Partitioned matroid:
  a matroid with a partition of the element set.
- Branch-width of a partitioned matroid:
  treat each part as a single element.

Then recursively find a pair a, b such that merging them does not increase branch-width. Merge them in one part and repeat.

Essence of the algorithm

From a given partitioned matroid $(M, \mathcal{P})$
represented over a finite field F,

- find a 'normalized matroid' N such that $\text{bwd}(M, \mathcal{P}) = \text{bwd}(N)$.
- Try to apply Hliněný's algorithm to
  decide whether branch-width of $N \leq k$.
- Attach a gadget to each part to create N.
- Make sure that N is representable over a finite filed F',
  where $|F'| <$ some function$(|F|, k)$.

Essence of the algorithm

From a given partitioned matroid $(M, \mathcal{P})$
represented over a finite field F,

- find a 'normalized matroid' N such that $\text{bwd}(M, \mathcal{P}) = \text{bwd}(N)$.
- Try to apply Hliněný's algorithm to
  decide whether branch-width of $N \leq k$.

- Attach a gadget to each part to create N.
- Make sure that N is representable over a finite filed F',
  where $|F'| <$ some function$(|F|, k)$.

Gadget: titanic set

Definition

- A set A is titanic if
  for every partition (X_1, X_2, X_3) of A,
  $\exists i$, $f(X_i) \geq f(A)$.
- A partition $\{P_1, P_2, \ldots, P_m\}$ is titanic
  if P_i is titanic for all i.
- Width of a partition: $\max f(P_i)$.

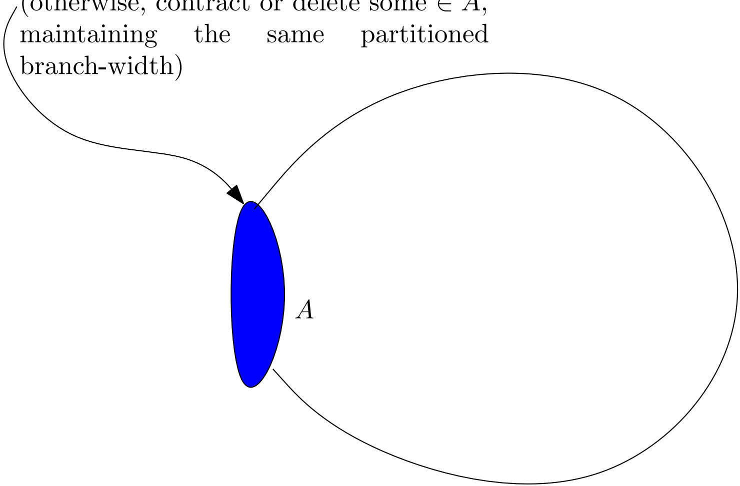
RS1991, Graph Minors X: if $\text{bwd}(f) \leq k$, $f(A) \leq k$, and A is titanic, then $V \setminus A$ is k-branched.

Theorem

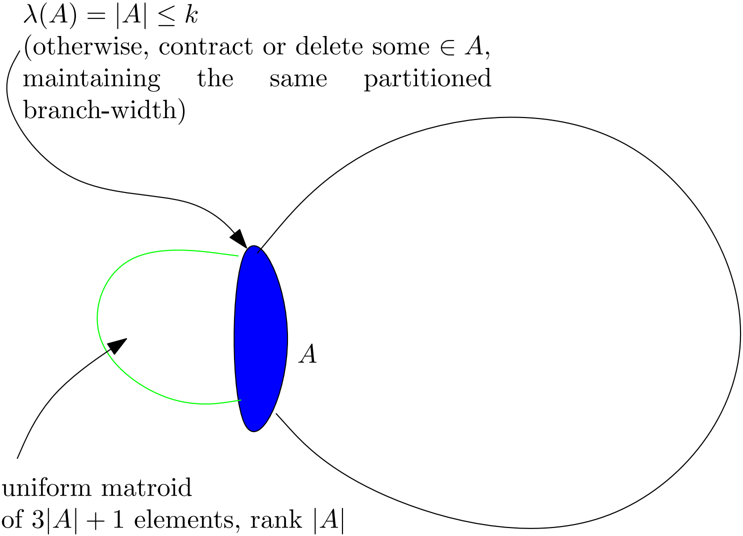
If $\mathcal{P}$: titanic partition of width $\leq k$, and $\text{bwd}(f) \leq k$,
then $\text{bwd}(f, \mathcal{P}) \leq k$.

Gadget for matroids: Amalgam with uniform matroids

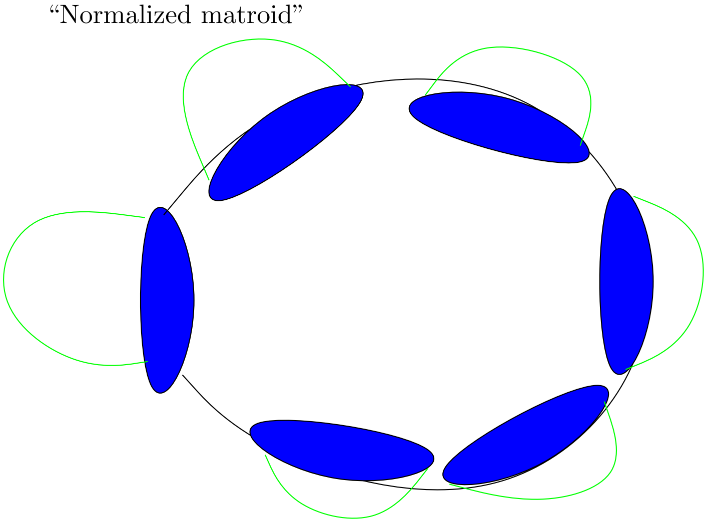
$\lambda(A) = |A| \leq k$
(otherwise, contract or delete some $\in A$, maintaining the same partitioned branch-width)



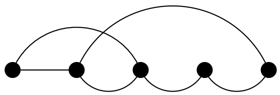
Gadget for matroids: Amalgam with uniform matroids

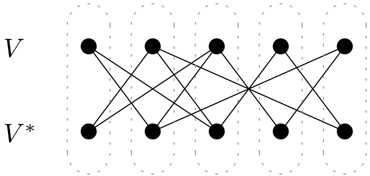


Gadget for matroids: Amalgam with uniform matroids



Graphs to Binary matroids





$$M = \text{matroid represented by } \begin{array}{cc} & \begin{array}{cc} V & V^* \end{array} \\ V & \left(\begin{array}{ccc|c} 1 & & & \\ & \ddots & & \text{Adjacency} \\ & & & \text{Matrix of } G \\ & & 1 & \end{array} \right) \end{array}.$$

Partition $\mathcal{P} = \{v, v^* : v \in V(G)\}$.

Rank-width of G = (Branch-width of $(M, \mathcal{P})$)/2

Running time

We can output

- branch-decomposition of matroids (represented over a fixed finite field) of width $\leq k$
- rank-decomposition of graphs of width $\leq k$

in time

- $O(n^6)$ with the naive implementation.
- $O(n^3)$ if combined Hliněný's algorithm more *seriously*.

(n: number of elements in a matroid, or number of vertices in a graph)

Can you do this for arbitrary connectivity functions?

Running time

We can output

- branch-decomposition of matroids (represented over a fixed finite field) of width $\leq k$
- rank-decomposition of graphs of width $\leq k$

in time

- $O(n^6)$ with the naive implementation.
- $O(n^3)$ if combined Hliněný's algorithm more *seriously*.

(n: number of elements in a matroid, or number of vertices in a graph)

Can you do this for arbitrary connectivity functions?

Running time

We can output

- branch-decomposition of matroids (represented over a fixed finite field) of width $\leq k$
- rank-decomposition of graphs of width $\leq k$

in time

- $O(n^6)$ with the naive implementation.
- $O(n^3)$ if combined Hliněný's algorithm more *seriously*.

(n: number of elements in a matroid, or number of vertices in a graph)

Can you do this for arbitrary connectivity functions?

Thanks for the attention!