Recognizing Rank-width

Sang-il Oum

Program in Applied and Computational Mathematics
Princeton University

Jan. 2005
Oberwolfach

Partially joint work with
Paul Seymour (Princeton University) and
Bruno Courcelle (LaBRI).

Outline

1. Introduction
   - Motivation
   - Introduction to Rank-width
2. Algorithm
   - Approximation Algorithm
   - Well-quasi-ordering
   - Decision Algorithm
3. Open problems

Treewidth vs Clique-width

Treewidth	Clique-width
Robertson and Seymour	Courcelle and Olariu
If twd $\leq k$, every MS_2 formula is decidable in linear time	If cwd $\leq k$, every MS_1 formula is decidable in polynomial time.
H is a minor of G $\Rightarrow \text{twd}(H) \leq \text{twd}(G)$.	H is an induced s.g. of G $\Rightarrow \text{cwd}(H) \leq \text{cwd}(G)$.
Large tree-width $\Leftrightarrow$ large grid minor	?
twd $\leq k$: linear time for fixed k.	cwd $\leq k$ is open for fixed $k > 3$.

Treewidth vs Clique-width

Treewidth	Clique-width
Robertson and Seymour	Courcelle and Olariu
If twd $\leq k$, every MS_2 formula is decidable in linear time	If cwd $\leq k$, every MS_1 formula is decidable in polynomial time.
H is a minor of G $\Rightarrow \text{twd}(H) \leq \text{twd}(G)$.	H is an induced s.g. of G $\Rightarrow \text{cwd}(H) \leq \text{cwd}(G)$.
Large tree-width $\Leftrightarrow$ large grid minor	?
twd $\leq k$: linear time for fixed k.	cwd $\leq k$ is open for fixed $k > 3$.

Treewidth vs Clique-width

Treewidth	Clique-width
Robertson and Seymour	Courcelle and Olariu
If twd $\leq k$, every MS_2 formula is decidable in linear time	If cwd $\leq k$, every MS_1 formula is decidable in polynomial time.
H is a minor of G $\Rightarrow \text{twd}(H) \leq \text{twd}(G)$.	H is an induced s.g. of G $\Rightarrow \text{cwd}(H) \leq \text{cwd}(G)$.
Large tree-width $\Leftrightarrow$ large grid minor	?
twd $\leq k$: linear time for fixed k.	cwd $\leq k$ is open for fixed $k > 3$.

Recognizing Tree-width at most k

Approximation Algorithm

Find a tree decomposition of G of width $\leq f(k)$ or confirm that tree-width $> k$.

Decision Algorithm

Using a tree decomposition of width $\leq f(k)$, decide whether tree-width $\leq k$.

- Well-quasi-ordering theorem of graphs of bounded tree-width implies that
  $\exists G_1, G_2, \ldots, G_{h(k)}$ such that
  $\mathrm{twd}(G) \leq k$ iff G_i is not isomorphic to any minor of G.
- For fixed graph H, we can test whether G contains an isomorphic copy of H as a minor in polynomial time if G is given by its tree decomposition.

Recognizing Tree-width at most k

Approximation Algorithm

Find a tree decomposition of G of width $\leq f(k)$ or confirm that tree-width $> k$.

Decision Algorithm

Using a tree decomposition of width $\leq f(k)$, decide whether tree-width $\leq k$.

- Well-quasi-ordering theorem of graphs of bounded tree-width implies that
  $\exists G_1, G_2, \ldots, G_{h(k)}$ such that
  $\mathrm{twd}(G) \leq k$ iff G_i is not isomorphic to any minor of G.
- For fixed graph H, we can test whether G contains an isomorphic copy of H as a minor in polynomial time if G is given by its tree decomposition.

Recognizing Tree-width at most k

Approximation Algorithm

Find a tree decomposition of G of width $\leq f(k)$ or confirm that tree-width $> k$.

Decision Algorithm

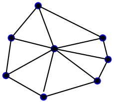
Using a tree decomposition of width $\leq f(k)$, decide whether tree-width $\leq k$.

- Well-quasi-ordering theorem of graphs of bounded tree-width implies that
  $\exists G_1, G_2, \ldots, G_{h(k)}$ such that
  $\mathrm{twd}(G) \leq k$ iff G_i is not isomorphic to any minor of G.
- For fixed graph H, we can test whether G contains an isomorphic copy of H as a minor in polynomial time if G is given by its tree decomposition.

Definition of Rank-width

O. and Seymour

- Rank-decomposition of G: a pair (T, L)
  - T: subcubic tree,
  - L: bijection from $V(G)$ to leaves of T.
- For each edge $e \in E(T)$, width of e
  - $= \mathrm{cutrk}_G(A_e)$
    where (A_e, B_e) is a partition of $V(G)$ given by $T \setminus e$.
- width of (T, L) = maximum width of e over $e \in E(T)$.
- Rank-width of G
  - Minimum width of Rank-decompositions of G.

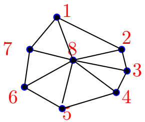


G

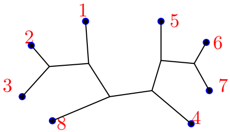
Definition of Rank-width

O. and Seymour

- Rank-decomposition of G: a pair (T, L)
  - T: subcubic tree,
  - L: bijection from $V(G)$ to leaves of T.
- For each edge $e \in E(T)$, width of e
  - $= \text{cutrk}_G(A_e)$
    where (A_e, B_e) is a partition of $V(G)$ given by $T \setminus e$.
- width of (T, L) = maximum width of e over $e \in E(T)$.
- Rank-width of G
  - Minimum width of Rank-decompositions of G.



G

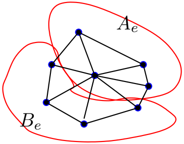


Rank-decomposition of G

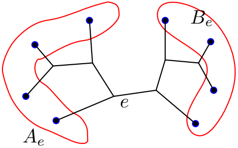
Definition of Rank-width

O. and Seymour

- Rank-decomposition of G: a pair (T, L)
  - T: subcubic tree,
  - L: bijection from $V(G)$ to leaves of T.
- For each edge $e \in E(T)$, width of e
  - $= \text{cutrk}_G(A_e)$
    where (A_e, B_e) is a partition of $V(G)$ given by $T \setminus e$.
- width of (T, L) = maximum width of e over $e \in E(T)$.
- Rank-width of G
  - Minimum width of Rank-decompositions of G.



G



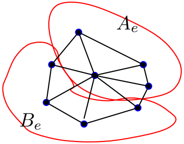
Rank-decomposition of G

$$\text{rank}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

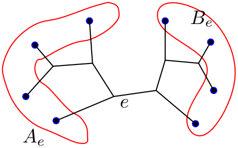
Definition of Rank-width

O. and Seymour

- Rank-decomposition of G: a pair (T, L)
  - T: subcubic tree,
  - L: bijection from $V(G)$ to leaves of T.
- For each edge $e \in E(T)$, width of e
  - $= \text{cutrk}_G(A_e)$
    where (A_e, B_e) is a partition of $V(G)$ given by $T \setminus e$.
- width of (T, L) = maximum width of e over $e \in E(T)$.
- Rank-width of G
  - Minimum width of Rank-decompositions of G.



G

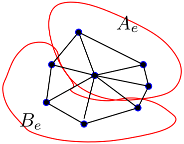


Rank-decomposition of G

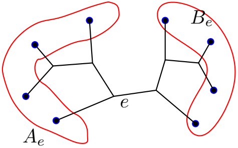
$$\text{rank}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Definition of Rank-width
O. and Seymour

- Rank-decomposition of G: a pair (T, L)
  - T: subcubic tree,
  - L: bijection from $V(G)$ to leaves of T.
- For each edge $e \in E(T)$, width of e
  - $= \mathrm{cutrk}_G(A_e)$
    where (A_e, B_e) is a partition of $V(G)$ given by $T \setminus e$.
- width of (T, L) = maximum width of e over $e \in E(T)$.
- Rank-width of G
  - Minimum width of Rank-decompositions of G.



G



Rank-decomposition of G

$$\operatorname{rank}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cut-rank

Definition of Cut-rank function cutrk : $2^{V(G)} \to \mathbb{Z}$

$\mathrm{cutrk}_G(A) = \mathrm{rank}(M)$,
M is a $A \times (V(G) \setminus A)$ matrix over $\mathbb{Z}_2$ such that

$$M_{xy} = \begin{cases} 1 & \text{if } xy \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

1. If M has at most k distict rows, then $\mathrm{rank}(M) \le k$. Conversely, if $\mathrm{rank}(M) = k$, then there are at most 2^k distinct rows.
2. Submodular inequality of a rank function

$$\mathrm{rank}\begin{pmatrix} A & B \\ C & D \end{pmatrix} + \mathrm{rank}(C) \ge \mathrm{rank}\begin{pmatrix} A \\ C \end{pmatrix} + \mathrm{rank}\begin{pmatrix} C & D \end{pmatrix}.$$

$$\Rightarrow \mathrm{cutrk}_G(X) + \mathrm{cutrk}_G(Y) \ge \mathrm{cutrk}_G(X \cap Y) + \mathrm{cutrk}_G(X \cup Y).$$

Cut-rank

Definition of Cut-rank function cutrk : $2^{V(G)} \to \mathbb{Z}$

$\text{cutrk}_G(A) = \text{rank}(M)$,
M is a $A \times (V(G) \setminus A)$ matrix over $\mathbb{Z}_2$ such that

$$M_{xy} = \begin{cases} 1 & \text{if } xy \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

1. If M has at most k distict rows, then $\text{rank}(M) \leq k$. Conversely, if $\text{rank}(M) = k$, then there are at most 2^k distinct rows.
2. Submodular inequality of a rank function

$$\text{rank}\begin{pmatrix} A & B \\ C & D \end{pmatrix} + \text{rank}(C) \geq \text{rank}\begin{pmatrix} A \\ C \end{pmatrix} + \text{rank}\begin{pmatrix} C & D \end{pmatrix}.$$

$$\Rightarrow \text{cutrk}_G(X) + \text{cutrk}_G(Y) \geq \text{cutrk}_G(X \cap Y) + \text{cutrk}_G(X \cup Y).$$

Properties of Rank-width

- $\mathrm{rwd}(G) \le \mathrm{cwd}(G) \le 2^{\mathrm{rwd}(G)+1} - 1$.
- $\mathrm{rwd}(G) \le 1$ iff G is distance-hereditary i.e. in every induced subgraph H and $u, v \in V(H)$, $d_H(u,v) = d_G(u,v)$.
- $\mathrm{rwd}(G \setminus v) = \mathrm{rwd}(G) - 1$ or $\mathrm{rwd}(G)$.
  $\mathrm{rwd}(G \setminus e) - \mathrm{rwd}(G) = 0, 1$, or -1.
  $\mathrm{rwd}(\overline{G}) - \mathrm{rwd}(G) = 0, 1$, or -1.
- $\mathrm{rwd}(G \oplus H) = \max(\mathrm{rwd}(G), \mathrm{rwd}(H))$.
- Robertson and Seymour (Graph Minors. X. '91)

Tangle Lemma

$\exists$ tangle of order $k \iff \mathrm{rwd} \ge k$.

This can be used to show that $\mathrm{rwd} \le k$ is co-NP.

Properties of Rank-width

- $\mathrm{rwd}(G) \le \mathrm{cwd}(G) \le 2^{\mathrm{rwd}(G)+1} - 1$.
- $\mathrm{rwd}(G) \le 1$ iff G is distance-hereditary i.e. in every induced subgraph H and $u, v \in V(H)$, $d_H(u,v) = d_G(u,v)$.
- $\mathrm{rwd}(G \setminus v) = \mathrm{rwd}(G) - 1$ or $\mathrm{rwd}(G)$.
  $\mathrm{rwd}(G \setminus e) - \mathrm{rwd}(G) = 0, 1,$ or -1.
  $\mathrm{rwd}(\overline{G}) - \mathrm{rwd}(G) = 0, 1,$ or -1.
- $\mathrm{rwd}(G \oplus H) = \max(\mathrm{rwd}(G), \mathrm{rwd}(H))$.
- Robertson and Seymour (Graph Minors. X. '91)

Tangle Lemma

$\exists$ tangle of order $k \iff \mathrm{rwd} \ge k$.

This can be used to show that $\mathrm{rwd} \le k$ is co-NP.

Properties of Rank-width

- $\mathrm{rwd}(G) \leq \mathrm{cwd}(G) \leq 2^{\mathrm{rwd}(G)+1} - 1$.
- $\mathrm{rwd}(G) \leq 1$ iff G is distance-hereditary i.e. in every induced subgraph H and $u, v \in V(H)$, $d_H(u, v) = d_G(u, v)$.
- $\mathrm{rwd}(G \setminus v) = \mathrm{rwd}(G) - 1$ or $\mathrm{rwd}(G)$.
  $\mathrm{rwd}(G \setminus e) - \mathrm{rwd}(G) = 0, 1$, or -1.
  $\mathrm{rwd}(\overline{G}) - \mathrm{rwd}(G) = 0, 1$, or -1.
- $\mathrm{rwd}(G \oplus H) = \max(\mathrm{rwd}(G), \mathrm{rwd}(H))$.
- Robertson and Seymour (Graph Minors. X. '91)

Tangle Lemma

$\exists$ tangle of order $k \iff \mathrm{rwd} \geq k$.

This can be used to show that $\mathrm{rwd} \leq k$ is co-NP.

Approximating Rank-width
O. and Seymour

Approx. Algorithm

For fixed k, there is a fixed-parameter-tractable algorithm that

- confirms that rank-width$> k$, or
- outputs the rank-decomposition of width $\leq 3k + 1$.

Running time: $O(n^9 \log n)$.
We use a general submodular function minimization algorithm $O(n^8 \log n)$ to do the following:

- Input: graph G, disjoint subsets $X, Y \subseteq V(G)$
- Output: Z, $X \subseteq Z \subseteq V(G) \setminus X$, that minimizes $\mathrm{cutrk}_G(Z)$.

Is there a faster or more direct algorithm?

Well-quasi-ordering of Graphs of Bounded Treewidth

Theorem (Robertson and Seymour)

If $\{G_1, G_2, \ldots\}$ is an infinite sequence of graphs of twd $\leq k$, then there exist $i < j$ such that G_i is isomorphic to a minor of G_j.

Corollary

For each k, $\exists$ list of graphs G_1, G_2, ..., $G_{h(k)}$ such that twd$(G) \leq k$ iff G_i is not isomorphic to a minor of G for all i.

We prove a similar statement for rank-width.

Local Complementation

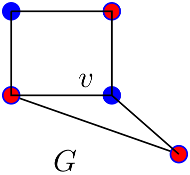
Local complementation at v

For all distinct neighbors x, y of v,
if $xy \in E(G)$, then remove the edge xy otherwise add an edge xy.

Let $G * v$ be a graph obtained by local complementation at v.

Cut-rank and Local Complementation

$\mathrm{cutrk}_G(X) = \mathrm{cutrk}_{G*v}(X)$.



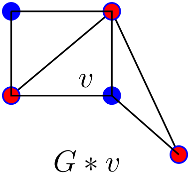
$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ & A & & & & B & & \\ & C & & & & D & & \end{pmatrix}$$

Local Complementation

Local complementation at v

For all distinct neighbors x, y of v,
if $xy \in E(G)$, then remove the edge xy otherwise add an edge xy.

Let $G * v$ be a graph obtained by local complementation at v.



Cut-rank and Local Complementation

$\mathrm{cutrk}_G(X) = \mathrm{cutrk}_{G*v}(X)$.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \multicolumn{4}{c}{\boxed{A}} & \multicolumn{4}{c}{B} \\ \multicolumn{4}{c}{C} & \multicolumn{4}{c}{D} \end{pmatrix}$$

Local Complementation

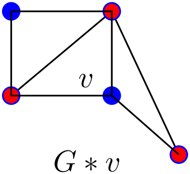
Local complementation at v

For all distinct neighbors x, y of v,
if $xy \in E(G)$, then remove the edge xy otherwise add an edge xy.

Let $G * v$ be a graph obtained by local complementation at v.

Cut-rank and Local Complementation

$\text{cutrk}_G(X) = \text{cutrk}_{G*v}(X)$.



$G * v$

$$\begin{pmatrix} 1 \quad 1 \quad 1 \quad 1 & 0 \quad 0 \quad 0 \quad 0 \\ \boxed{A} & B \\ C & D \end{pmatrix}$$

Vertex-minor

Definition

H is a vertex-minor of G if H can be obtained from G by applying a sequence of

- vertex deletions and
- local complementations.

Then, if H is a vertex-minor of G, then $rwd(H) \leq \mathrm{rwd}(G)$.
For an edge uv of G, let $G \wedge uv = G * u * v * u$, called a pivoting. Note that $G * u * v * u = G * v * u * v$.

Definition

H is a pivot-minor of G if H can be obtained from G by applying a sequence of *vertex deletions* and *pivotings*.

Minor of Binary matroid $M \Leftrightarrow$ Pivot-minor of a fundamental graph of M.

Well-quasi-ordering of Graphs of Bounded Rank-width

Theorem

If $\{G_1, G_2, \ldots\}$ is an infinite sequence of graphs of rwd $\leq k$, then there exist $i < j$ such that

G_i is isomorphic to a pivot-minor/vertex-minor of G_j.

Proof ideas:

1. Isotropic system by A. Bouchet
2. Similar to the proof of well-quasi-ordering of representable matroids over a finite field of bounded branch-width by Geelen, Gerards, and Whittle. (Actually, our theorem implies their theorem for binary matroids.)

Forbidden Vertex-Minors

Corollary

For each k, $\exists$ list of graphs $G_1, G_2, \ldots, G_{h(k)}$ such that $\mathrm{rwd}(G) \leq k$ iff G_i is not isomorphic to a vertex-minor of G for all i.

- This corollary has an elementary proof saying that $|V(G_i)| \leq (6^{k+1} - 1)/5$.
- Is there a polynomimal-time algorithm to test whether an input graph contains a fixed graph as a vertex-minor (or pivot-minor)? Open.

Checking a Fixed Vertex-minor

Courcelle and O.

- Let H be a fixed graph.
- We construct a C_2MS_1 formula φ_H that describes whether H is isomorphic to a vertex-minor of G.
  Main idea
  - (A. Bouchet)
    vertex-minor of graphs $\Leftrightarrow$ minor of isotropic systems.
  - Logical formulation of isotropic systems.
- By the previous corollary, we obtain a C_2MS_1 formula φ_k that decides whether $\mathrm{rwd}(G) \leq k$.
- (Courcelle) Every C_2MS_1 formula on G is decidable in polynomial time if $\mathrm{cwd}(G) \leq k$ for a fixed k.

Combining Everything

Recognizing rwd $\leq k$

Run the approximation algorithm. $O(n^9 \log n)$ time.

- If it outputs that rwd $> k$ and stop.
- Otherwise, we obtain the rank-decomposition of width at most $3k + 1$.
  - Convert it into the $(2^{3k+2} - 1)$-*expression* related to clique-width. $O(n^2)$ time.
  - Use it to test a C_2MS_1 formula describing that rwd $\leq k$. $O(n)$ time.

Combining Everything

Recognizing rwd $\leq k$

Run the approximation algorithm. $O(n^9 \log n)$ time.

- If it outputs that rwd $> k$ and stop.
- Otherwise, we obtain the rank-decomposition of width at most $3k+1$.
  - Convert it into the $(2^{3k+2}-1)$-*expression* related to clique-width. $O(n^2)$ time.
  - Use it to test a C_2MS_1 formula describing that rwd $\leq k$. $O(n)$ time.

Combining Everything

Recognizing rwd $\leq k$

Run the approximation algorithm. $O(n^9 \log n)$ time.

- If it outputs that rwd $> k$ and stop.
- Otherwise, we obtain the rank-decomposition of width at most $3k + 1$.
  - Convert it into the $(2^{3k+2} - 1)$-*expression* related to clique-width. $O(n^2)$ time.
  - Use it to test a C_2MS_1 formula describing that rwd $\leq k$. $O(n)$ time.

Combining Everything

Recognizing rwd $\leq k$

Run the approximation algorithm. $O(n^9 \log n)$ time.

- If it outputs that rwd $> k$ and stop.
- Otherwise, we obtain the rank-decomposition of width at most $3k + 1$.
  - Convert it into the $(2^{3k+2} - 1)$-*expression* related to clique-width. $O(n^2)$ time.
  - Use it to test a C_2MS_1 formula describing that rwd $\leq k$. $O(n)$ time.

Combining Everything

Recognizing rwd $\leq k$

Run the approximation algorithm. $O(n^9 \log n)$ time.

- If it outputs that rwd $> k$ and stop.
- Otherwise, we obtain the rank-decomposition of width at most $3k+1$.
  - Convert it into the $(2^{3k+2}-1)$-*expression* related to clique-width. $O(n^2)$ time.
  - Use it to test a C_2MS_1 formula describing that rwd $\leq k$. $O(n)$ time.

Open Problems

1. For a fixed k, is it possible to construct the rank-decomposition of width at most k, *if there is one*, in polynomial time?
2. Can we avoid using the general submodular minimization algorithm?
   - Let A, B be disjoint subsets of $V(G)$. Can we find a polynomial-time algorithm to find Z minimizing $\text{cutrk}_G(Z)$ such that $A \subseteq Z \subseteq V(G) \setminus B$?
     If G is bipartite, this can be done by the Matroid intersection Theorem.
3. When does a graph have large rank-width (or clique-width)?
4. Is the rank-width of $n \times n$ grid $n - 1$?