Approximating Rank-width and Clique-width Quickly

Sang-il Oum

Program in Applied and Computational Mathematics
Princeton University

June 23, 2005

31st International Workshop
on Graph-Theoretic Concepts in Computer Science

Part I

Introduction to Rank-width and Clique-width

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

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

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

Clique-width vs Rank-width

- The definition of clique-width is not so convenient.
- We define another complexity measure of graphs called rank-width,
- If rank-width is small, then clique-width is small and vice versa.

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

- We use cut-rank functions to define rank-width.

cut-rank

G: graph, A: adjacency matrix of G over $\mathrm{GF}(2)$.
Let $A[X, Y]$ denote a submatrix of A with rows$= X$, columns$= Y$.

$$\rho_G(X) = \text{rank}(A[X, V(G) \setminus X])$$

Clique-width vs Rank-width

- The definition of clique-width is not so convenient.
- We define another complexity measure of graphs called rank-width,
- If rank-width is small, then clique-width is small and vice versa.

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

- We use cut-rank functions to define rank-width.

cut-rank

G: graph, A: adjacency matrix of G over $\mathrm{GF}(2)$.
Let $A[X, Y]$ denote a submatrix of A with rows$= X$, columns$= Y$.

$$\rho_G(X) = \text{rank}(A[X, V(G) \setminus X])$$

Clique-width vs Rank-width

- The definition of clique-width is not so convenient.
- We define another complexity measure of graphs called rank-width,
- If rank-width is small, then clique-width is small and vice versa.

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

- We use cut-rank functions to define rank-width.

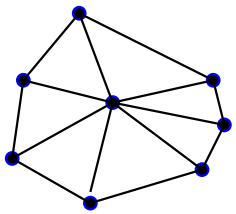
cut-rank

G: graph, A: adjacency matrix of G over $\mathrm{GF}(2)$.
Let $A[X, Y]$ denote a submatrix of A with rows$= X$, columns$= Y$.

$$\rho_G(X) = \text{rank}(A[X, V(G) \setminus X])$$

Rank-width
(with Seymour)

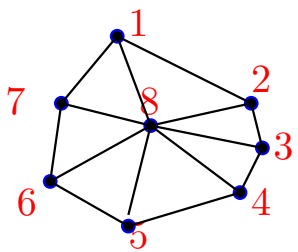
- Rank-decomposition of G: a pair $(T, \mathcal{L})$
  - T: subcubic tree,
  - $\mathcal{L}$: bijection from $V(G)$ to leaves of T.
- For each edge $e \in E(T)$, width of e
  - $= \rho_G(A_e)$
    where (A_e, B_e) is a partition of $V(G)$ given by $T \setminus e$.
- width of $(T, \mathcal{L})$ = maximum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



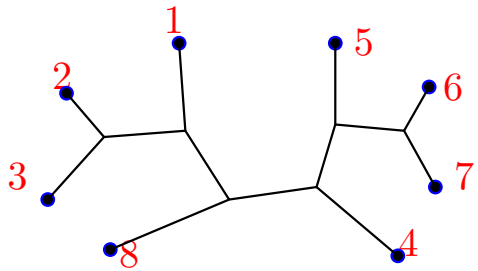
G

Rank-width
(with Seymour)

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



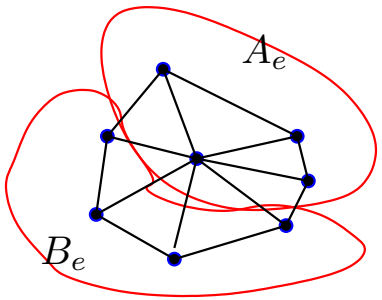
G



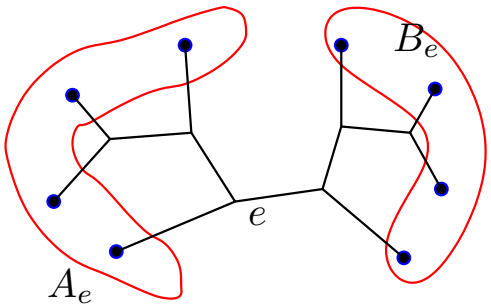
Rank-decomposition of G

Rank-width
(with Seymour)

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



G

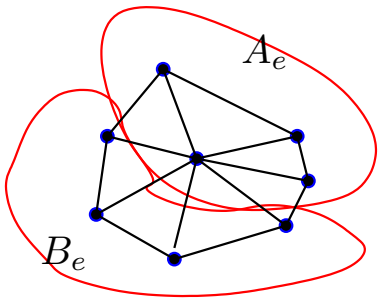


Rank-decomposition of G

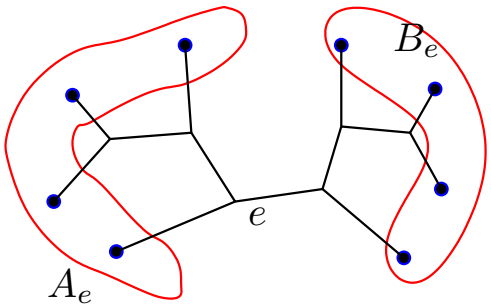
$$\text{rank} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rank-width
(with Seymour)

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



G

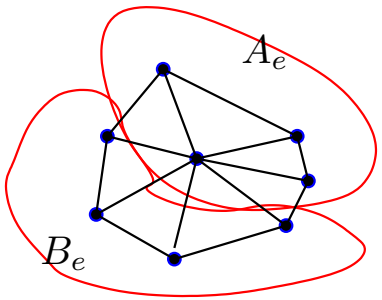


Rank-decomposition of G

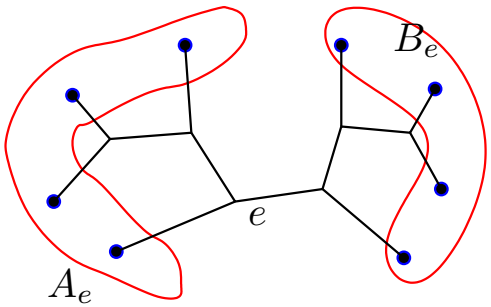
$$\text{rank} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rank-width
(with Seymour)

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

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

Rank-width: Why interesting?

Classical excuse: (cartoon from Garey and Johnson's book)

"I can't find an efficient algorithm for this graph problem, but neither can all these famous people."

Rank-width: Why interesting?

“How about restricting inputs?
I can find an efficient algorithm
if input graphs are restricted to graphs of small rank-width.”

Algorithms based on Small Clique-width

- Hamiltoian path/circuit Wanke (1994), Espelage, Gurski, and Wanke (2001), Finding the chromatic number Kobler and Rotics (2003), Maximum weight stable set, ...
- All graph problems, expressible in monadic second-order logic formulas with quantifications over vertices and vertex sets. Courcelle, Makowsky, and Rotics (2000)
  (logic formulas with $\neg$, $\vee$, $\wedge$, (,), $x = y$, $x \sim y$, $x \in X$, $\forall x$, $\exists y$, $\forall X$, $\exists Y$)

Algorithms based on Small Clique-width

- Hamiltoian path/circuit Wanke (1994), Espelage, Gurski, and Wanke (2001), Finding the chromatic number Kobler and Rotics (2003), Maximum weight stable set, ...
- All graph problems, expressible in monadic second-order logic formulas with quantifications over vertices and vertex sets. Courcelle, Makowsky, and Rotics (2000)
  (logic formulas with $\neg$, $\vee$, $\wedge$, (,), $x = y$, $x \sim y$, $x \in X$, $\forall x$, $\exists y$, $\forall X$, $\exists Y$)

"Frankly, ... most of them require its k-expression as an input.
I don't know how to find it even if it is known to exist."

Approximating Rank-width: 3 Algorithms

Approximating Rank-width

Poly-time algorithm that either outputs a rank-decomp. of width $\leq f(k)$ or confirms that rank-width $> k$.

1. (Previous result with Seymour) $f(k) = 3k + 1$, $O(n^9 \log n)$
   Uses the submodular function minimization alg.
2. $f(k) = 3k + 1$, $O(n^4)$
   We find a faster minimization algorithm for cut-rank functions.
3. $f(k) = 24k$, $O(n^3)$
   Reduce the problem to branch-width of binary matroids. Then use Hliněný's algorithm.

Approximating Rank-width: 3 Algorithms

Approximating Rank-width

Poly-time algorithm that either outputs a rank-decomp. of width $\leq f(k)$ or confirms that rank-width $> k$.

1. (Previous result with Seymour) $f(k) = 3k + 1$, $O(n^9 \log n)$
   Uses the submodular function minimization alg.
2. $f(k) = 3k + 1$, $O(n^4)$
   We find a faster minimization algorithm for cut-rank functions.
3. $f(k) = 24k$, $O(n^3)$
   Reduce the problem to branch-width of binary matroids. Then use Hliněný's algorithm.

Approximating Rank-width: 3 Algorithms

Approximating Rank-width

Poly-time algorithm that either outputs a rank-decomp. of width $\leq f(k)$ or confirms that rank-width $> k$.

1. (Previous result with Seymour) $f(k) = 3k + 1$, $O(n^9 \log n)$
   Uses the submodular function minimization alg.
2. $f(k) = 3k + 1$, $O(n^4)$
   We find a faster minimization algorithm for cut-rank functions.
3. $f(k) = 24k$, $O(n^3)$
   Reduce the problem to branch-width of binary matroids. Then use Hliněný's algorithm.

Part II

$O(n^4)$-time algorithm

Speed up the previous algorithm

Seymour and Oum (2004) use the submodular function minimization algorithm (that runs in $O(n^8 \log n)$) to solve the following:

Minimizing Cut-rank

Let k: constant.
For disjoint subsets X, Y of $V(G)$, $|X|, |Y| \leq k$,
find Z, $X \subseteq Z \subseteq V(G) \setminus Z$, minimizing $\rho_G(Z)$.

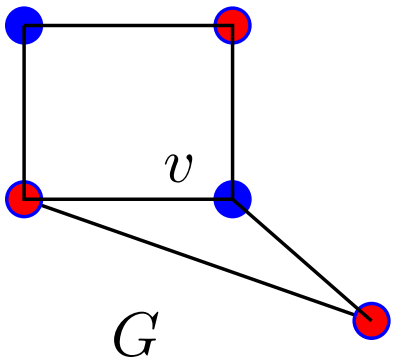
We find $O(n^3)$-algorithm for minimizing cut-rank.
This will make the previous algorithm run in $O(n^4)$ time.

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.

$G * v$: graph obtained by local complementation at v.

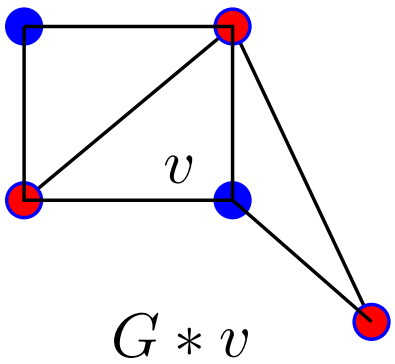


Local Complementation

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.

$G * v$: graph obtained by local complementation at v.

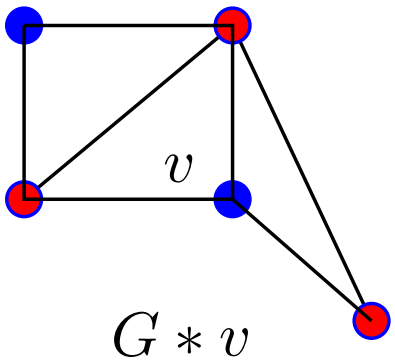


Local Complementation

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.

$G * v$: graph obtained by local complementation at v.



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

$\rho_G(X) = \rho_{G*v}(X)$.

Moreover, a pivoting on bipartite graph is equal to $G * v * w * v$.

Idea of minimizing the cut-rank function

Let A be the adjacency matrix of a graph G.

- Cut-rank functions are preserved under pivoting.
- rank $A[X, Y]$ is a lower bound of $\rho_G(Z)$.

Keep pivoting until rank $A[X, Y]$ reaches its maximum.

Blocking sequences: technique developed by Jim Geelen 1995.
If rank $A[X, Y]$ is not maximum, there is a blocking sequence that explains why.

1. Find a blocking sequence. (Shortest path problem on the auxiliary digraph.)
2. If the sequence has length m, a sequence of m pivoting increases rank $A[X, Y]$ at least by one.
3. Repeat. (at most k times because maximum is at most k.)

Complexity of minimizing cut-rank functions

- Constructing the auxiliary digraph: $O(n^2)$.
- Finding a blocking sequence: $O(n^2)$.
- Pivoting $O(n^2)$.
  We do pivoting at most n times at each step.

$O(k(n^2 + n^2 + n \times n^2)) = O(n^3)$ for minimizing cut-rank functions.

$O(n^4)$-time algorithm that
either outputs a rank-decomp. of width $\leq 3k + 1$
or confirms that rank-width $> k$

$O(n^4)$-time algorithm that
either outputs a $(2^{3k+1} - 1)$-expression
or confirms that clique-width $> k$

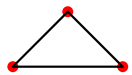
Part III

$O(n^3)$-time algorithm

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

$B(G)$

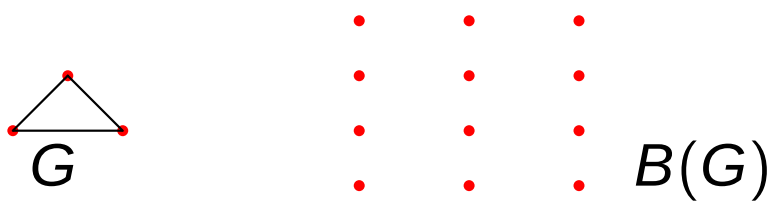
Theorem

If $E(G) \neq \emptyset$, then
$\frac{1}{4}$ rank-width of $G \leq$ rank-width of $B(G) \leq 2$ rank-width of G.

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.



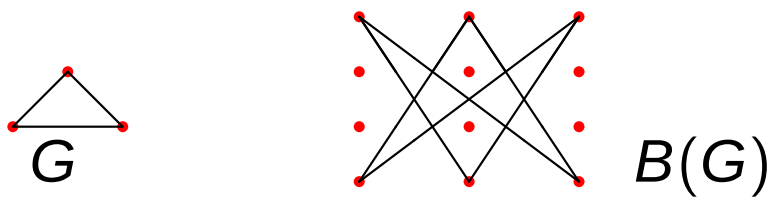
Theorem

If $E(G) \neq \emptyset$, then
$\frac{1}{4}$ rank-width of $G \leq$ rank-width of $B(G) \leq 2$ rank-width of G.

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.



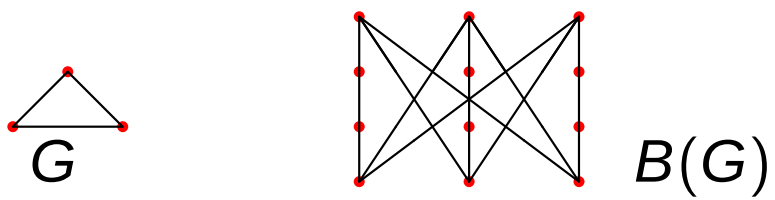
Theorem

If $E(G) \neq \emptyset$, then
$\frac{1}{4}$ rank-width of $G \leq$ rank-width of $B(G) \leq 2$ rank-width of G.

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.



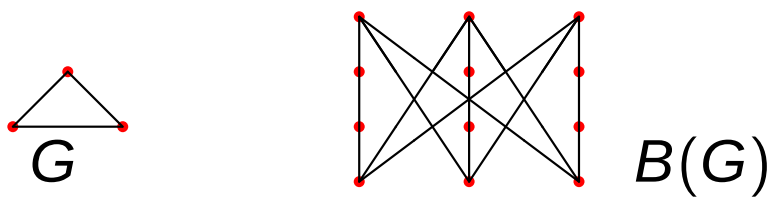
Theorem

If $E(G) \neq \emptyset$, then
$\frac{1}{4}$ rank-width of $G \leq$ rank-width of $B(G) \leq 2$ rank-width of G.

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
$\frac{1}{4}$ rank-width of $G \leq$ rank-width of $B(G) \leq 2$ rank-width of G.

Bipartite graphs and Rank-width

Theorem (Oum 2004)

Rank-width of a Bipartite graph G
= (Branch-width of the Binary Matroid obtained from G) -1.

Theorem (Hliněný 2002)

(Originally for matroids representable over a finite field.)
For bipartite graphs G, there is $O(n^3)$-time algorithm that
either outputs the rank-decomposition of width $\leq 3k'$
or proves that rank-width $> k'$.

We give $B(G)$ to the input of Hliněný's algorithm with $k' = 4k$.

For graphs G, there is $O(n^3)$-time algorithm that
either outputs the rank-decomposition of width $\leq 24k$
or proves that rank-width $> k$.

Part IV

Discussions

Implication to Algorithms based on Small Clique-width

1. We have a $O(n^2)$-time algorithm transforming a rank-decomposition of width k into the $(2^{k+1}-1)$-expression.

   width $24k \rightarrow$
   $(2 \cdot 16777216^k - 1)$-expession.
2. This can be given as an input to algorithms based on small clique-width.
3. Since k is constant, we no longer require k-expressions as an input to obtain the polynomial-time algorithms.
   Many linear-time algorithms requiring k-expressions are now running in $O(n^3)$ time without requiring the k-expressions.

Approximating Rank-width

$O(n^3)$-time algorithm that either outputs a rank-decomp. of width $\leq 24k$ or confirms that rank-width $> k$.

$n = |V(G)|$.

Implication to Algorithms based on Small Clique-width

1. We have a $O(n^2)$-time algorithm transforming a rank-decomposition of width k into the $(2^{k+1}-1)$-expression.

   width $24k \rightarrow$
   $(2 \cdot 16777216^k - 1)$-expession.
2. This can be given as an input to algorithms based on small clique-width.
3. Since k is constant, we no longer require k-expressions as an input to obtain the polynomial-time algorithms.
   Many linear-time algorithms requiring k-expressions are now running in $O(n^3)$ time without requiring the k-expressions.

Approximating Rank-width

$O(n^3)$-time algorithm that either outputs a rank-decomp. of width $\leq 24k$ or confirms that rank-width $> k$.

$n = |V(G)|$.

Implication to Algorithms based on Small Clique-width

1. We have a $O(n^2)$-time algorithm transforming a rank-decomposition of width k into the $(2^{k+1}-1)$-expression.

   width $24k \rightarrow$
   $(2 \cdot 16777216^k - 1)$-expession.
2. This can be given as an input to algorithms based on small clique-width.
3. Since k is constant, we no longer require k-expressions as an input to obtain the polynomial-time algorithms.
   Many linear-time algorithms requiring k-expressions are now running in $O(n^3)$ time without requiring the k-expressions.

Approximating Rank-width

$O(n^3)$-time algorithm that either outputs a rank-decomp. of width $\leq 24k$ or confirms that rank-width $> k$.

$n = |V(G)|$.

Open problems

Construction problem

Can you find a rank-decomposition of width $\leq k$ if there is one in polynomial time? (k:fixed)
It is solved that the decision problem (rank-width$\leq k$) is in P.

Thank you for your attention!

Acknowledgement

- Jim Geelen: suggested the use of blocking sequences at the Oberwolfach workshop in Jan. 2005.
- Bruno Courcelle: $B(G)$.
- Organizers!