Twin-width

Édouard Bonnet
based on joint works with Colin Geniet, Eun Jung Kim,
Stéphan Thomassé, and Rémi Watrigant

ENS Lyon, LIP

January 20th, 2021, tutorial at Universität Bremen
Graphs

Two outcomes between a pair of vertices: edge or non-edge
Trigraphs

Three outcomes between a pair of vertices: edge, or non-edge, or red edge (error edge)
Contractions in trigraphs

Identification of two non-necessarily adjacent vertices
Contractions in trigraphs

Identification of two non-necessarily adjacent vertices
Contractions in trigraphs

edges to $N(u) \triangle N(v)$ turn red, for $N(u) \cap N(v)$ red is absorbing
A contraction sequence of $G$:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$. 
A contraction sequence of $G$:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$. 
A contraction sequence of $G$:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$.
A contraction sequence of $G$:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$.
A contraction sequence of $G$:

Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$. 

Contraction sequence

\[
\begin{align*}
\text{c} & \quad \text{bef} \\
\text{bef} & \quad \text{adg}
\end{align*}
\]
A contraction sequence of $G$:

Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$. 
A contraction sequence of $G$:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that $G_i$ is obtained by performing one contraction in $G_{i+1}$.
Twin-width

tww(G): Least integer $d$ such that $G$ admits a contraction sequence where all trigraphs have *maximum red degree* at most $d$.

![Graph](image)

*Maximum red degree $= 0$*

*overall maximum red degree $= 0$*
Twin-width

tww($G$): Least integer $d$ such that $G$ admits a contraction sequence where all trigraphs have maximum red degree at most $d$.

Maximum red degree = 2
overall maximum red degree = 2
Twin-width

tww\((G)\): Least integer \(d\) such that \(G\) admits a contraction sequence where all trigraphs have \textit{maximum red degree} at most \(d\).

Maximum red degree = 2
overall maximum red degree = 2
Twin-width

tww(G): Least integer $d$ such that $G$ admits a contraction sequence where all trigraphs have maximum red degree at most $d$.

Maximum red degree $= 2$

overall maximum red degree $= 2$
Twin-width

tww($G$): Least integer $d$ such that $G$ admits a contraction sequence where all trigraphs have *maximum red degree* at most $d$.

Maximum red degree = 1
overall maximum red degree = 2
Twin-width

tww\((G)\): Least integer \(d\) such that \(G\) admits a contraction sequence where all trigraphs have \textit{maximum red degree} at most \(d\).

![Diagram showing bcef and adg with maximum red degree = 1, overall maximum red degree = 2.]

Maximum red degree = 1
overall maximum red degree = 2
Twin-width

tww\( (G) \): Least integer \( d \) such that \( G \) admits a contraction sequence where all trigraphs have maximum red degree at most \( d \).

Maximum red degree = 0
overall maximum red degree = 2
Simple operations preserving small twin-width

- complementation: remains the same
- taking induced subgraphs: may only decrease
- adding one vertex linked arbitrarily: at most “doubles”
Complementation

\[ \text{tww}(\overline{G}) = \text{tww}(G) \]
Complementation

\[ \overline{G_6} \]

\[ G_6 \]

\[ \text{tww}(\overline{G}) = \text{tww}(G) \]
Induced subgraph

\[ \text{tww}(H) \leq \text{tww}(G) \]
Induced subgraph

Ignoring absent vertices
Induced subgraph

Mimic the contractions otherwise
Induced subgraph

Mimic the contractions otherwise
Induced subgraph

Mimic the contractions otherwise
Induced subgraph

\[ \text{Mimic the contractions otherwise} \]
Induced subgraph

Mimic the contractions otherwise
Adding one vertex $v$

Left as an exercise

**Hint:** Up until the very end, $v$ shall have no incident red edge
Graphs with bounded twin-width – trees

If possible, contract two twin leaves
Graphs with bounded twin-width – trees

If not, contract a deepest leaf with its parent
Graphs with bounded twin-width – trees

If not, contract a deepest leaf with its parent
Graphs with bounded twin-width – trees

If possible, contract two twin leaves
Graphs with bounded twin-width – trees

Cannot create a red degree-3 vertex
Graphs with bounded twin-width – trees

Cannot create a red degree-3 vertex
Graphs with bounded twin-width – trees

Cannot create a red degree-3 vertex
Graphs with bounded twin-width – trees

Cannot create a red degree-3 vertex
Graphs with bounded twin-width – trees

Cannot create a red degree-3 vertex
Graphs with bounded twin-width – trees

Cannot create a red degree-3 vertex
Graphs with bounded twin-width – trees

Generalization to bounded \textit{treewidth} and even bounded \textit{rank-width}
Graphs with bounded twin-width – grids
Graphs with bounded twin-width – grids
Graphs with bounded twin-width – grids
Graphs with bounded twin-width – grids
Graphs with bounded twin-width – grids
Graphs with bounded twin-width – grids
Graphs with bounded twin-width – grids

4-sequence for planar grids, 3$d$-sequence for $d$-dimensional grids
Universal bipartite graph

No $O(1)$-contraction sequence:

twin-width is not an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:

**twin-width is *not* an iterated identification of near twins.**
Universal bipartite graph

No $O(1)$-contraction sequence: twin-width is \emph{not} an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is not an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:

twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is \textit{not} an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:

**twin-width is not** an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:
twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence:

* twin-width is *not* an iterated identification of near twins.
Universal bipartite graph

No $O(1)$-contraction sequence: twin-width is not an iterated identification of near twins.
Graphs with bounded twin-width – planar graphs?
Graphs with bounded twin-width – planar graphs?

For every $d$, a planar trigraph without planar $d$-contraction
Graphs with bounded twin-width – planar graphs?

For every $d$, a planar trigraph without planar $d$-contraction

More powerful tool needed
Twin-width in the language of matrices

Encode a bipartite graph (or, if symmetric, any graph)
Twin-width in the language of matrices

Contraction of two columns (similar with two rows)
Twin-width in the language of matrices

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & r & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & r & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & r & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & r & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

How is the twin-width (re)defined?
Twin-width in the language of matrices

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & \_ & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & \_ & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & \_ & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

How to tune it for non-bipartite graph?
Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are *consecutive*

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]
Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are consecutive

Maximum number of non-constant zones per column or row part
\[\begin{array}{cccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array}\]

= error value
Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are *consecutive*

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Maximum number of non-constant zones per column or row part
... until there are a single row part and column part
Partition viewpoint

Matrix partition: partitions of the row set and of the column set
Matrix division: same but all the parts are consecutive

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Twin-width as maximum error value of a contraction/division sequence
**Grid minor**

A matrix is said \( t \)-grid free if it does not have a \( t \times t \)-division where every cell is non-empty.

Non-empty cell: contains at least one 1 entry

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

4-grid minor
Grid minor

*t-grid minor*: $t \times t$-division where every cell is non-empty

Non-empty cell: contains at least one 1 entry

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

4-grid minor

A matrix is said *t-grid free* if it does not have a *t*-grid minor
Mixed minor

Mixed cell: not horizontal nor vertical

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

3-mixed minor
Mixed minor

Mixed cell: not horizontal nor vertical

```
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 \\
\end{array}
```

3-mixed minor

Every mixed cell is witnessed by a $2 \times 2$ square = \textbf{corner}
Mixed minor

Mixed cell: not horizontal nor vertical

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
\end{bmatrix}
\]

3-mixed minor

A matrix is said \textit{t-mixed free} if it does not have a \textit{t}-mixed minor
Mixed value

\[
\begin{array}{cccc|ccc}
R_4 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\hline
R_3 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
& 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
\hline
R_2 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
& 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
R_1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
& 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[\approx \text{(maximum) number of cells with a corner per row/column part}\]
### Mixed value

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_2$</td>
<td>0 1 1 0 0</td>
<td>1 0 0 0 1</td>
<td>1 0 1 0 0</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>0 1 0 0 1</td>
<td>1 0 0 1 0</td>
<td>1 0 1 0 0</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>1 0 1 0 0</td>
<td>0 0 0 0 1</td>
<td>1 0 1 0 0</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>1 0 1 0 0</td>
<td>0 0 0 0 1</td>
<td>1 0 1 0 0</td>
<td>1 0 1 0 0</td>
</tr>
</tbody>
</table>

But we add the number of *boundaries* containing a corner
Mixed value

\[
\begin{array}{cccc|cc}
R_4 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
\hline
\cup & & & & & & & \\
R_3 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
\hline
R_2 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
\hline
\end{array}
\]

\[
\therefore \text{merging row parts do not increase mixed value of column part}
\]
Twin-width and mixed freeness

**Theorem**

*If* $G$ *admits a* $t$-*mixed free adjacency matrix, then* $tww(G) = 2^{O(t)}$. 

---

**Step 1:** find a division sequence $(D_i)_{i=1}^n$ with mixed value $f(t)$.
Twin-width and mixed freeness

Theorem

If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{2^{O(t)}}$. 

Step 1: find a division sequence $(D_i)$ with mixed value $f(t)$

Merge parts greedily
Twin-width and mixed freeness

Theorem
If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{2^O(t)}$.

Step 1: find a division sequence $(D_i)_i$ with mixed value $f(t)$

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Merge consecutive parts greedily
Twin-width and mixed freeness

Theorem
If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{O(t)}$.

Step 1: find a division sequence $(\mathcal{D}_i)_i$ with mixed value $f(t)$

<table>
<thead>
<tr>
<th>1 1 1 1 1 1 1 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 0 1 0 1 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>0 1 0 0 1 0 1 0 0</td>
</tr>
<tr>
<td>1 0 0 1 1 0 1 0 0</td>
</tr>
<tr>
<td>0 1 1 1 1 1 1 0 0</td>
</tr>
<tr>
<td>1 0 1 1 1 1 0 0 1</td>
</tr>
</tbody>
</table>

Merge consecutive parts greedily
Twin-width and mixed freeness

**Theorem**

If \( \exists \sigma \) s.t. \( \text{Adj}_\sigma(G) \) is \( t \)-mixed free, then \( \text{tww}(G) = 2^{O(t)} \).

**Step 1: find a division sequence \((D_i)_i\) with mixed value \( f(t) \)**

\[
\begin{array}{ccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

Merge consecutive parts greedily
Twin-width and mixed freeness

**Theorem**

If \( \exists \sigma \) s.t. \( Adj_\sigma(G) \) is \( t \)-mixed free, then \( \text{tww}(G) = 2^{2^{O(t)}} \).

**Step 1:** find a division sequence \( (D_i)_i \) with mixed value \( f(t) \)

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

Stuck, removing every other separation \( \rightarrow \frac{f(t)}{2} \) mixed cells per part
Stanley-Wilf conjecture / Marcus-Tardos theorem

Auxiliary 0,1-matrix with one entry per cell: a 1 iff the cell is mixed

Question

For every $k$, is there a $c_k$ such that every $n \times m$ 0,1-matrix with at least $c_k$ 1 per row and column admits a $k$-grid minor?
Stanley-Wilf conjecture / Marcus-Tardos theorem

Auxiliary 0,1-matrix with one entry per cell: a 1 iff the cell is mixed

Conjecture (reformulation of Füredi-Hajnal conjecture ’92)

For every $k$, there is a $c_k$ such that every $n \times m$ 0,1-matrix with at least $c_k \max(n, m)$ 1 entries admits a $k$-grid minor.
Stanley-Wilf conjecture / Marcus-Tardos theorem

Auxiliary 0,1-matrix with one entry per cell: a 1 iff the cell is mixed

Conjecture (reformulation of Füredi-Hajnal conjecture ’92)
For every k, there is a $c_k$ such that every $n \times m$ 0,1-matrix with at least $c_k \max(n, m)$ 1 entries admits a $k$-grid minor.

Conjecture (Stanley-Wilf conjecture ’80s)
Any proper permutation class contains only $2^{O(n)}$ $n$-permutations.

Klazar showed Füredi-Hajnal $\Rightarrow$ Stanley-Wilf in 2000
Marcus and Tardos showed Füredi-Hajnal in 2004
Let $M$ be an $n \times n$ 0,1-matrix without $k$-grid minor
Marcus-Tardos one-page inductive proof

\[ M = \begin{array}{cccc}
\hline
\hline
\hline
\hline
\hline
\end{array} \]

\[ k^2 \times k^2 \]

Draw a regular \( \frac{n}{k^2} \times \frac{n}{k^2} \) division on top of \( M \)
Marcus-Tardos one-page inductive proof

\[ M = \begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & 1 & \ldots \\ 1 & 1 & 1 & 1 \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix} \]

\( k^2 \times k^2 \)

A cell is *wide* if it has at least \( k \) columns with a 1.
Marcus-Tardos one-page inductive proof

A cell is *tall* if it has at least $k$ rows with a 1
There are less than $k\binom{k^2}{k}$ wide cells per column part. Why?
Marcus-Tardos one-page inductive proof

There are less than $k \binom{k^2}{k}$ tall cells per row part
Marcus-Tardos one-page inductive proof

\[
M = \begin{bmatrix}
  & W & & & \\
  W & W & W & T & \\
  T & W & T & T & T \\
  T & & & & \\
  k^2 \times k^2 & & & & \\
  & W & & & \\
\end{bmatrix}
\]

In W and T, at most
\[
2 \cdot \frac{n}{k^2} \cdot k^{k^2} \cdot k^4 = 2 k^3 \binom{k^2}{k} n
\]
entries 1.
There are at most \((k - 1)^2 c_k \frac{n}{k^2}\) remaining 1. Why?
Choose \( c_k = 2k^4 \binom{k^2}{k} \) so that \((k - 1)^2 c_k \frac{n}{k^2} + 2k^3 \binom{k^2}{k} n \leq c_k n \)
Twin-width and mixed freeness

**Theorem**

If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{O(t)}$.

**Step 1: find a division sequence $(D_i)_i$ with mixed value $f(t)$**

![Division sequence matrix]

Stuck, removing every other separation $\rightarrow \frac{f(t)}{2}$ mixed cells per part
Twin-width and mixed freeness

Theorem

If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{O(t)}$.

Step 1: find a division sequence $(D_i)_i$ with mixed value $f(t)$

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
\end{bmatrix}
\]

Stuck, removing every other separation $\rightarrow \frac{f(t)}{2}$ mixed cells per part

Impossible!
Twin-width and mixed freeness

Theorem

If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{O(t)}$.

Step 1: find a division sequence $(D_i)_i$ with mixed value $f(t)$

Step 2: find a contraction sequence with error value $g(t)$

Refinement of $D_i$ where each part coincides on the non-mixed cells
Twin-width and mixed freeness

Theorem

If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{O(t)}$. 
Twin-width and mixed freeness

Theorem
If $\exists \sigma$ s.t. $\text{Adj}_\sigma(G)$ is $t$-mixed free, then $\text{tww}(G) = 2^{2^{O(t)}}$.

Now to bound the twin-width of a class $C$:
1) Find a good vertex-ordering procedure
2) Argue that, in this order, a $t$-mixed minor would conflict with $C$
Unit interval graphs

Intersection graph of unit segments on the real line
Bounded twin-width – unit interval graphs

order by left endpoints
Bounded twin-width – unit interval graphs

No 3-by-3 grid has all 9 cells crossed by two non-decreasing curves
Graph minors

Formed by **vertex deletion**, **edge deletion**, and **edge contraction**

A graph $G$ is $H$-minor free if $H$ is not a minor of $G$

A graph class is $H$-minor free if all its graphs are
Graph minors

Formed by **vertex deletion, edge deletion, and edge contraction**

A graph $G$ is $H$-minor free if $H$ is not a minor of $G$

A graph class is $H$-minor free if all its graphs are

Planar graphs are exactly the graphs without $K_5$ or $K_{3,3}$ as a minor

[Diagram of $K_5$ and $K_{3,3}$]
Bounded twin-width – $K_t$-minor free graphs

Given a hamiltonian path, we would just use this order
Bounded twin-width – $K_t$-minor free graphs

Contracting the $2t$ subpaths yields a $K_{t,t}$-minor, hence a $K_t$-minor
Bounded twin-width – $K_t$-minor free graphs

Instead we use a specially crafted lex-DFS discovery order
Theorem
The following classes have bounded twin-width, and $O(1)$-sequences can be computed in polynomial time.

- Bounded rank-width, and even, boolean-width graphs,
- Every hereditary proper subclass of permutation graphs,
- Posets of bounded antichain size (seen as digraphs),
- Unit interval graphs,
- $K_t$-minor free graphs,
- Map graphs,
- Subgraphs of $d$-dimensional grids,
- $K_t$-free unit $d$-dimensional ball graphs,
- $\Omega(\log n)$-subdivisions of all the $n$-vertex graphs,
- Cubic expanders defined by iterative random 2-lifts from $K_4$,
- Strong products of two bounded twin-width classes, one with bounded degree, etc.
Theorem
The following classes have bounded twin-width, and $O(1)$-sequences can be computed in polynomial time.

- Bounded rank-width, and even, boolean-width graphs,
- every hereditary proper subclass of permutation graphs,
- posets of bounded antichain size (seen as digraphs),
- unit interval graphs,
- $K_t$-minor free graphs,
- map graphs,
- subgraphs of $d$-dimensional grids,
- $K_t$-free unit $d$-dimensional ball graphs,
- $\Omega(\log n)$-subdivisions of all the $n$-vertex graphs,
- cubic expanders defined by iterative random 2-lifts from $K_4$,
- strong products of two bounded twin-width classes, one with bounded degree, etc.

Can we solve problems faster, given an $O(1)$-sequence?
Cographs

A single vertex is a cograph,
Cographs

\[ G_1 \cup \alpha(G_1) + \alpha(G_2) \]

as well as the union of two cographs,
Cographs

\[ G_1 \cup \alpha(G_1) + \alpha(G_2) \]

and the complete join of two cographs.
Many NP-hard problems are polytime solvable on cographs
Cographs

Let’s try to compute the NP-hard $\alpha(G)$, independence number.
Cographs

In case of a disjoint union: combine the solutions
Cographs

\[ G_1 \cup \alpha(G_1) + \alpha(G_2) \quad \max\{\alpha(G_1), \alpha(G_2)\} \]

In case of a complete join: pick the larger one
Cographs

\[ G_1 \cup \alpha(G_1) + \alpha(G_2) \]

In case of a complete join: pick the larger one

\[ \max\{\alpha(G_1), \alpha(G_2)\} \]
Equivalent cograph definition

Cographs form the unique maximal hereditary class in which every\(^1\) graph has two *twins*

\(^1\)provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique maximal hereditary class in which every\(^1\) graph has two *twins* ...wait a minute

\(^1\)provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique \textit{maximal hereditary} class in which every\footnote{provided it has at least two vertices} graph has two \textit{twins} \dots yes, they coincide with \texttt{twin-width 0}
Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every\(^1\) graph has two *twins* ...yes, they coincide with *twin-width* 0

![Diagram of a cograph]

Is there another algorithmic scheme based on this definition?

\(^1\)provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique maximal hereditary class in which every\(^1\) graph has two twins \(...\)yes, they coincide with \textbf{twin-width 0}

Let's try with \(\alpha(G)\), and store in a vertex its inner max solution

\(^1\)provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique maximal hereditary class in which every \(^1\) graph has two *twins* ...yes, they coincide with **twin-width 0**

![Diagram of cographs]

We can find a pair of false/true twins

---

\(^1\) provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique *maximal hereditary* class in which every\(^1\) graph has two *twins* ...yes, they coincide with **twin-width 0**

Sum them if they are false twins

\(^1\)provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique maximal hereditary class in which every\(^1\) graph has two twins ...yes, they coincide with twin-width 0

Max them if they are true twins

\(^1\)provided it has at least two vertices
Equivalent cograph definition

Cographs form the unique \textit{maximal hereditary} class in which every\footnote{provided it has at least two vertices} graph has two \textit{twins} ...yes, they coincide with \textbf{twin-width 0}

Why does it eventually compute $\alpha(G)$?
Example of $k$-Independent Set

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: Compute by dynamic programming a best partial solution in each red connected subgraph of size at most $k$. 
Example of $k$-Independent Set

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: Compute by dynamic programming a best partial solution in each red connected subgraph of size at most $k$. 
$d^{2k}n^2$ red connected subgraphs, actually only $d^{2k}n = 2^{O_d(k)}n$
Example of $k$-Independent Set

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: Compute by dynamic programming a best partial solution in each red connected subgraph of size at most $k$.

$d^{2k}n^2$ red connected subgraphs, actually only $d^{2k}n = 2^{O_d(k)}n$

In $G_n$: red connected subgraphs are singletons, so are the solutions.
In $G_1$: If solution of size at least $k$, global solution.
Example of $k$-Independent Set

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: Compute by dynamic programming a best partial solution in each red connected subgraph of size at most $k$.

$d^{2k}n^2$ red connected subgraphs, actually only $d^{2k}n = 2^{O_d(k)}n$

In $G_n$: red connected subgraphs are singletons, so are the solutions.
In $G_1$: If solution of size at least $k$, global solution.

How to go from the partial solutions of $G_{i+1}$ to those of $G_i$?
Best partial solution inhabiting $\bullet$?
3 unions of $\leq d + 2$ red connected subgraphs to consider in $G_{i+1}$ with $u$, or $v$, or both
Other (almost) single-exponential parameterized algorithms

**Theorem**

*Given a d-sequence \( G = G_n, \ldots, G_1 = K_1 \),

- \( k \)-Independent Set,
- \( k \)-Clique,
- \( (r, k) \)-Scattered Set,
- \( k \)-Dominating Set, and
- \( (r, k) \)-Dominating Set*

*can be solved in time \( 2^{O(k)} n \),

whereas Subgraph Isomorphism and Induced Subgraph Isomorphism *can be solved in time \( 2^{O(k \log k)} n \).*
Other (almost) single-exponential parameterized algorithms

Theorem
Given a d-sequence $G = G_n, \ldots, G_1 = K_1$,
- $k$-Independent Set,
- $k$-Clique,
- $(r, k)$-Scattered Set,
- $k$-Dominating Set, and
- $(r, k)$-Dominating Set can be solved in time $2^{O(k)} n$,

whereas Subgraph Isomorphism and Induced Subgraph Isomorphism can be solved in time $2^{O(k \log k)} n$.

A more general FPT algorithm?
First-order model checking on graphs

| Graph FO Model Checking | Parameter: $|\varphi|$ |
|-------------------------|-------------------------|
| **Input:** A graph $G$ and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$ |
| **Question:** $G \models \varphi$? |
First-order model checking on graphs

**Graph FO Model Checking**

**Input:** A graph $G$ and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$

**Question:** $G \models \varphi$?

**Parameter:** $|\varphi|$

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \forall x \bigvee_{1 \leq i \leq k} x = x_i \lor \bigvee_{1 \leq i \leq k} E(x, x_i) \lor E(x_i, x)$$

$G \models \varphi$? $\iff$
First-order model checking on graphs

**Graph FO Model Checking**

**Parameter:** $|\varphi|$  

**Input:** A graph $G$ and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$

**Question:** $G \models \varphi$?

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \forall x \bigvee_{1 \leq i \leq k} x = x_i \lor \bigvee_{1 \leq i \leq k} E(x, x_i) \lor E(x_i, x)$$

$G \models \varphi ? \iff k$-Dominating Set

$G \models \varphi$?
First-order model checking on graphs

**Parameter:** $|\varphi|$

**Input:** A graph $G$ and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$

**Question:** $G \models \varphi$?

**Example:**

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \wedge \bigwedge_{1 \leq i < j \leq k} \neg(x_i = x_j) \wedge \neg E(x_i, x_j) \wedge \neg E(x_j, x_i)$$

$G \models \varphi$? $\Leftrightarrow$
First-order model checking on graphs

**Graph FO Model Checking**

**Parameter:** $|\varphi|$

**Input:** A graph $G$ and a first-order sentence $\varphi \in FO(\{E_2, =_2\})$

**Question:** $G \models \varphi$?

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \bigwedge_{1 \leq i < j \leq k} \neg(x_i = x_j) \land \neg E(x_i, x_j) \land \neg E(x_j, x_i)$$

$G \models \varphi \iff k$-Independent Set
FO interpretations and transductions

**FO interpretation:** redefine the edges by a first-order formula

\[ \varphi(x, y) = \neg E(x, y) \quad \text{(complement)} \]

\[ \varphi(x, y) = E(x, y) \lor \exists z E(x, z) \land E(z, y) \quad \text{(square)} \]
FO interpretations and transductions

**FO interpretation:** redefine the edges by a first-order formula

\[ \varphi(x, y) = \neg E(x, y) \quad \text{(complement)} \]
\[ \varphi(x, y) = E(x, y) \lor \exists z E(x, z) \land E(z, y) \quad \text{(square)} \]

**FO transduction:** color by \( O(1) \) unary relations, interpret, delete

Theorem Bounded twin-width is preserved by transduction.
FO interpretations and transductions

**FO interpretation:** redefine the edges by a first-order formula
\[ \varphi(x, y) = \neg E(x, y) \] (complement)

\[ \varphi(x, y) = E(x, y) \lor \exists z E(x, z) \land E(z, y) \] (square)

**FO transduction:** color by \( O(1) \) unary relations, interpret, delete

\[
\begin{align*}
\text{Theorem:} & \quad \text{Bounded twin-width is preserved by transduction.}
\end{align*}
\]
FO interpretations and transductions

**FO interpretation:** redefine the edges by a first-order formula

\[ \varphi(x, y) = \neg E(x, y) \] (complement)

\[ \varphi(x, y) = E(x, y) \lor \exists z E(x, z) \land E(z, y) \] (square)

**FO transduction:** color by \( O(1) \) unary relations, interpret, delete

\[ \varphi(x, y) = E(x, y) \lor (G(x) \land B(y) \land \neg \exists z R(z) \land E(y, z)) \]

\[ \lor (R(x) \land B(y) \land \exists z R(z) \land E(y, z) \land \neg \exists z B(z) \land E(y, z)) \]
**FO interpretations and transductions**

**FO interpretation:** redefine the edges by a first-order formula

\[ \varphi(x, y) = \neg E(x, y) \]  
(complement)

\[ \varphi(x, y) = E(x, y) \lor \exists z E(x, z) \land E(z, y) \]  
(square)

**FO transduction:** color by \( O(1) \) unary relations, interpret, delete

\[ \varphi(x, y) = E(x, y) \lor (G(x) \land B(y) \land \neg \exists z R(z) \land E(y, z)) \]

\[ \lor (R(x) \land B(y) \land \exists z R(z) \land E(y, z) \land \neg \exists z B(z) \land E(y, z)) \]
FO interpretations and transductions

**FO interpretation:** redefine the edges by a first-order formula
\[ \varphi(x, y) = \neg E(x, y) \] (complement)
\[ \varphi(x, y) = E(x, y) \lor \exists zE(x, z) \land E(z, y) \] (square)

**FO transduction:** color by \( O(1) \) unary relations, interpret, delete

![Diagrams showing FO interpretations and transductions](image-url)
FO interpretations and transductions

**FO interpretation:** redefine the edges by a first-order formula

\[ \varphi(x, y) = \neg E(x, y) \] (complement)

\[ \varphi(x, y) = E(x, y) \lor \exists z E(x, z) \land E(z, y) \] (square)

**FO transduction:** color by \( O(1) \) unary relations, interpret, delete

---

**Theorem**

*Bounded twin-width is preserved by transduction.*
Monadically Stable and NIP

**Stable class:** no transduction of the class contains all ladders

**NIP class:** no transduction of the class contains all graphs

![Diagram](image)
Monadically Stable and NIP

**Stable class:** no transduction of the class contains all ladders

**NIP class:** no transduction of the class contains all graphs

Bounded-degree graphs $→$ stable
Unit interval graphs $→$ NIP but not stable
Interval graphs $→$ not NIP
Monadically Stable and NIP

**Stable class:** no transduction of the class contains all ladders

**NIP class:** no transduction of the class contains all graphs

Bounded-degree graphs $\rightarrow$ stable

Unit interval graphs $\rightarrow$ NIP but not stable

Interval graphs $\rightarrow$ not NIP

Bounded twin-width classes $\rightarrow$ NIP but not stable in general
Classes with known tractable FO model checking

- NIP \ stable
  - bounded rank-width
  - cographs
  - dense classes
  - posets of bounded width
  - L-interval
  - unit interval
  - pattern avoiding permutations

- nowhere dense
  - bounded expansion
  - polynomial expansion
  - proper minor-closed
  - map graphs
    - planar
  - bounded degree
  - “sparse” classes
  - stable
Classes with known tractable FO model checking

**FO Model Checking** solvable in $f(|\varphi|)n$ on bounded-degree graphs

[Seese ’96]
Classes with known tractable FO model checking

**FO Model Checking** solvable in $f(|\varphi|)n^{1+\varepsilon}$ on any nowhere dense class [Grohe, Kreutzer, Siebertz '14]
Classes with known tractable FO model checking

NIP \ stable

bounded rank-width

cographs

dense classes

bounded expansion

polynomial expansion

proper minor-closed

bounded twin-width

rank-width

cographs

L-interval

unit interval

pattern avoiding permutations

map graphs

planar

nowhere dense

bounded degree

“sparse” classes

stable

End of the story for the subgraph-closed classes

tractable FO Model Checking ⇔ nowhere dense ⇔ stable
Classes with known tractable FO model checking

NIP \ stable

- bounded rank-width
- cographs
- dense classes
- posets of bounded width
- \( L \)-interval
- unit interval
- pattern avoiding permutations

nowhere dense

- bounded expansion
- polynomial expansion
- proper minor-closed
- map graphs
- planar

stable

bounded degree

“sparse” classes

New program: transductions of nowhere dense classes
Not sparse anymore but still stable
Classes with known tractable FO model checking

- NIP \ stable

- bounded rank-width
  - cographs
  - posets of bounded width
  - L-interval
  - unit interval
  - pattern avoiding permutations

- nowhere dense
  - bounded expansion
  - polynomial expansion
  - proper minor-closed
  - map graphs
  - planar

- “sparse” classes

- bounded degree

**MSO\_1 Model Checking** solvable in $f(|\varphi|, w)n$ on graphs of rank-width $w$

[Courcelle, Makowsky, Rotics '00]
Classes with known tractable FO model checking

\[ \text{Is } \sigma \text{ a subpermutation of } \tau? \text{ solvable in } f(|\sigma||\tau|) \]

[Guillemot, Marx '14]
Classes with known tractable FO model checking

FO Model Checking solvable in $f(|\varphi|, w)n^2$ on posets of width $w$ [GHLOORS '15]
Classes with known tractable FO model checking

**FO Model Checking** solvable in $f(|\varphi|)n^{O(1)}$ on map graphs

[Eickmeyer, Kawarabayashi ’17]
Classes with known tractable FO model checking

FO Model Checking solvable in $f(|\varphi|, d)n$ on graphs with a $d$-sequence
Workflow of the FO model checking algorithm

- Graph $G$ of bounded twin-width
- $n^{O(1)}$
- $t$-mixed-free order
- $d$-contraction sequence $G = G_n, \ldots, G_1 = K_1$
- $n^{O(1)}$
- Reduced morphism-tree $MT'_\ell(G)$ of size $h(\ell)$
- $O(\ell, d(n))$
- Query $G \models \varphi$ for any prenex $\varphi$ of depth $\ell$
- $O(\ell(1))$

Direct examples: trees, bounded rank-width, grids, $d$-dimensional grids, $K_t$-free unit ball graphs
Workflow of the FO model checking algorithm

graph $G$ of bounded twin-width

$t$-mixed-free order

$d$-contraction sequence $G = G_n, \ldots, G_1 = K_1$

reduced morphism-tree $MT'_\ell(G)$ of size $h(\ell)$

Query $G \models \varphi$ for any prenex $\varphi$ of depth $\ell$

Direct examples: trees, bounded rank-width, grids, $d$-dimensional grids, $K_t$-free unit ball graphs
Workflow of the FO model checking algorithm

Graph $G$ of bounded twin-width $\rightarrow$ $t$-mixed-free order $\rightarrow$ $d$-contraction sequence $G = G_n, \ldots, G_1 = K_1$

Reduced morphism-tree $MT'_\ell(G)$ of size $h(\ell)$ $\rightarrow$ Query $G \models \varphi$ for any prenex $\varphi$ of depth $\ell$

Detour via mixed minor for: pattern-avoiding permutations, unit intervals, bounded width posets, $K_t$-minor free graphs
Workflow of the FO model checking algorithm

- **Graph** $G$ of bounded twin-width
  - Reduced morphism-tree $MT'_\ell(G)$ of size $h(\ell)$
  - Query $G \models \varphi$ for any prenex $\varphi$ of depth $\ell$

- $t$-mixed-free order
  - $n^{O(1)}$

- $d$-contraction sequence $G = G_n, \ldots, G_1 = K_1$
  - $n^{O(1)}$

Generalization of what we saw for $k$-**Independent Set**
Small classes

Small: class with at most $n!c^n$ labeled graphs on $[n]$.

**Theorem**

*Bounded twin-width classes are small.*

Unifies and extends the same result for:

- $\sigma$-free permutations [Marcus, Tardos ’04]
- $K_t$-minor free graphs [Norine, Seymour, Thomas, Wollan ’06]
Small classes

Small: class with at most $n!c^n$ labeled graphs on $[n]$.

**Theorem**

*Bounded twin-width classes are small.*

Subcubic graphs, interval graphs, triangle-free unit segment graphs have **unbounded** twin-width
Small classes

Small: class with at most $n!c^n$ labeled graphs on $[n]$.

**Theorem**

*Bounded twin-width classes are small.*

Is the converse true for hereditary classes?

**Conjecture (small conjecture)**

*A hereditary class has bounded twin-width if and only if it is small.*
\( \chi \)-boundedness

\[ \mathcal{C} \ \chi \text{-bounded: } \exists f, \forall G \in \mathcal{C}, \ \chi(G) \leq f(\omega(G)) \]

**Theorem**

*Every twin-width class is \( \chi \)-bounded.*

*More precisely, every graph \( G \) of twin-width at most \( d \) admits a proper \( (d + 2)^{\omega(G)-1} \)-coloring.*
\[\chi\text{-boundedness}\]

\[C \ \chi\text{-bounded: } \exists f, \forall G \in C, \ \chi(G) \leq f(\omega(G))\]

**Theorem**

*Every twin-width class is \(\chi\)-bounded.*

*More precisely, every graph \(G\) of twin-width at most \(d\) admits a proper \((d + 2)^{\omega(G) - 1}\)-coloring.*

Polynomially \(\chi\)-bounded? i.e., \(\chi(G) = O(\omega(G)^d)\)
$d + 2$-coloring in the triangle-free case

Algorithm: **Start from** $G_1 = K_1$, color its unique vertex 1, and rewind the $d$-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.
\(d + 2\)-coloring in the triangle-free case

Algorithm: **Start from** \(G_1 = K_1\), color its unique vertex 1, and rewind the \(d\)-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.

\[N_{G_i}[z]\]

\[N_{G_{i+1}}[u, v]\]

\(z\) has only red incident edges \(\rightarrow d + 2\)-nd color available to \(v\)
$d + 2$-coloring in the triangle-free case

Algorithm: Start from $G_1 = K_1$, color its unique vertex 1, and rewind the $d$-sequence. A contraction seen backward is a split and we shall find colors for the two new vertices.

$N_{G_i}[z] \rightarrow \text{non-edge between } u \text{ and } v$

$z$ incident to at least one black edge
Future directions

Main questions:
Algorithm to compute/approximate twin-width in general
Fully classify classes with tractable FO model checking
Small conjecture
Better approximation algorithms on bounded twin-width classes
Twin-width of Cayley graphs of finitely generated groups...
Future directions

Main questions:
Algorithm to compute/approximate twin-width in general
Fully classify classes with tractable FO model checking
Small conjecture
Better approximation algorithms on bounded twin-width classes
Twin-width of Cayley graphs of finitely generated groups...

On arxiv
Twin-width I: tractable FO model checking [BKTW '20]
Twin-width II: small classes [BGKTW '20]
Twin-width III: Max Independent Set and Coloring [BGKTW '20]