Factorising Pattern-Free Permutations

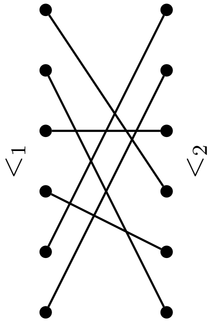
Édouard Bonnet   Romain Bourneuf
Colin Geniet   Stéphan Thomassé

ENS Lyon

SODA 24

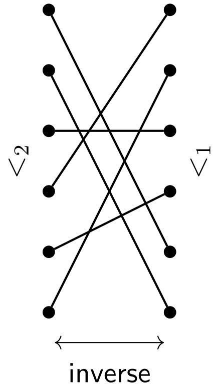
Permutations

Permutation = two linear orders on the same set: $(X, <_1, <_2)$



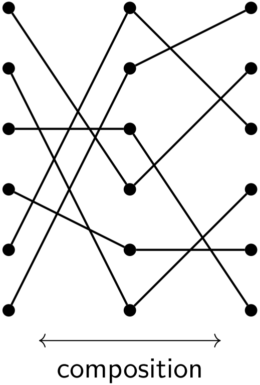
Permutations

Permutation = two linear orders on the same set: $(X, <_1, <_2)$

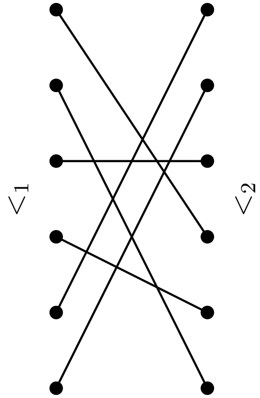


Permutations

Permutation = two linear orders on the same set: $(X, <_1, <_2)$

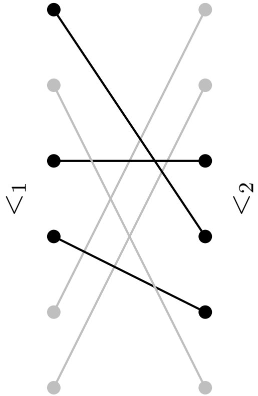


Patterns in permutations



Permutation $(X, <_1, <_2)$

Patterns in permutations

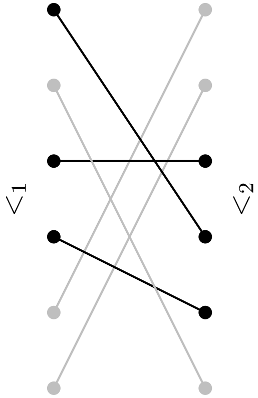


Permutation $(X, <_1, <_2)$

Pattern = induced substructure

$$(Y, <_1, <_2), \quad Y \subset X$$

Patterns in permutations



Permutation $(X, <_1, <_2)$

Pattern $=$ induced substructure

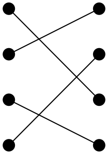
$$(Y, <_1, <_2), \quad Y \subset X$$

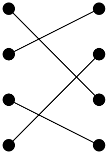
Pattern-free permutation class:

$$\mathcal{F}(\tau) = \{\sigma \ : \ \tau \not\subseteq \sigma\}$$

Separable permutations

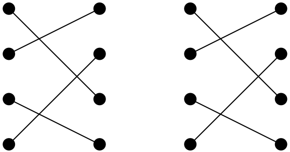
Separable permutations
$= \mathcal{F}(3142, 2413)$



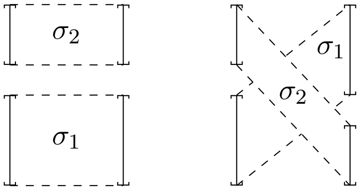


Separable permutations

Separable permutations
$= \mathcal{F}(3142, 2413)$

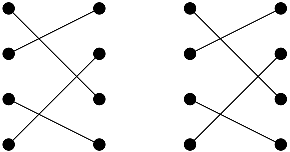


$=$ permutations created by direct/skew
sum

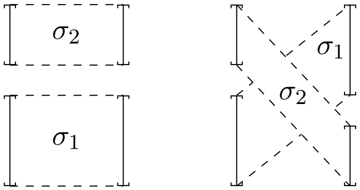


Separable permutations

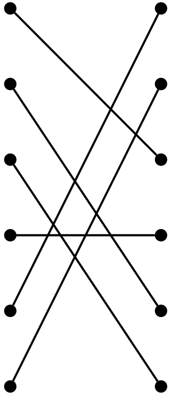
Separable permutations
$= \mathcal{F}(3142, 2413)$



$=$ permutations created by direct/skew sum

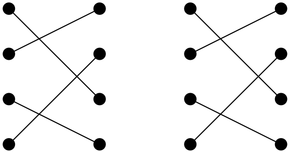


Tree representation:

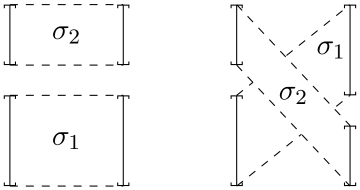


Separable permutations

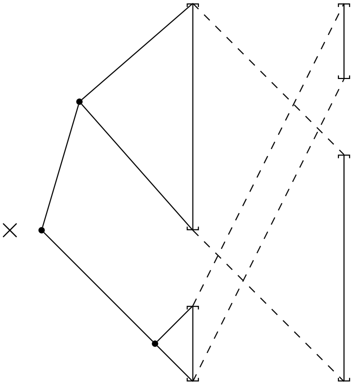
Separable permutations
$= \mathcal{F}(3142, 2413)$



= permutations created by direct/skew sum

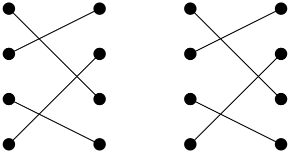


Tree representation:

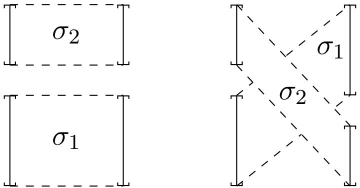


Separable permutations

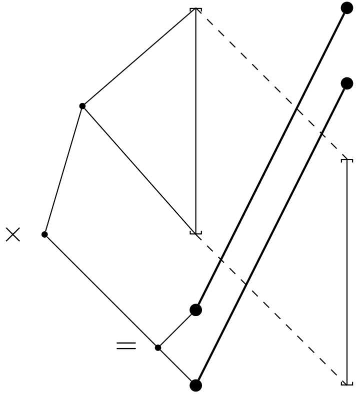
Separable permutations
$= \mathcal{F}(3142, 2413)$



$=$ permutations created by direct/skew sum

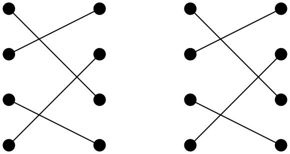


Tree representation:

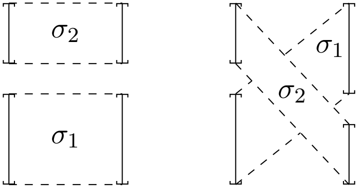


Separable permutations

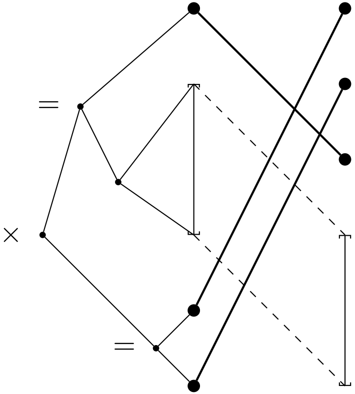
Separable permutations
$= \mathcal{F}(3142, 2413)$



$=$ permutations created by direct/skew sum

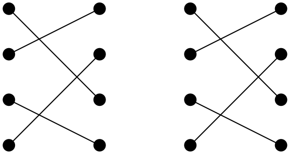


Tree representation:

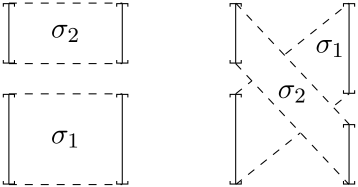


Separable permutations

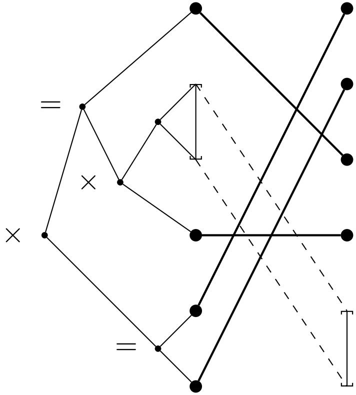
Separable permutations
$= \mathcal{F}(3142, 2413)$



$=$ permutations created by direct/skew sum

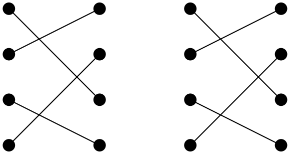


Tree representation:

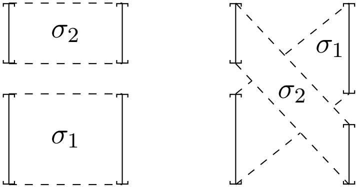


Separable permutations

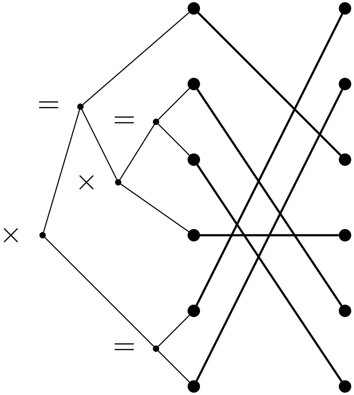
Separable permutations
$= \mathcal{F}(3142, 2413)$



$=$ permutations created by direct/skew sum



Tree representation:



Pattern-free classes are nice

Theorem (Marcus–Tardos '04)

For any τ, there is a constant c such that $\mathcal{F}(\tau)$ has $\leq c^n$ permutations of size n.

Pattern-free classes are nice

Theorem (Marcus–Tardos '04)

For any τ, there is a constant c such that $\mathcal{F}(\tau)$ has $\leq c^n$ permutations of size n.

Recognition algorithm:

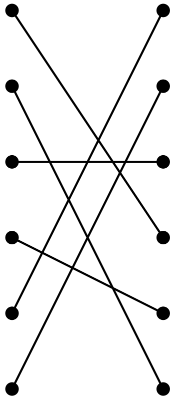
Theorem (Guillemot–Marx '14)

One can test if τ is a pattern of σ in time $f(\tau) \cdot |\sigma|$.

Twin-width

Twin-width of $(X, <_1, <_2)$:

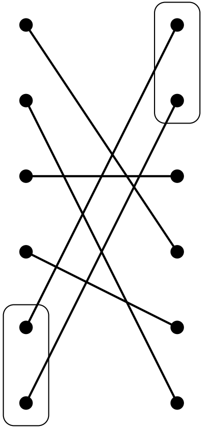
- iteratively merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree



Twin-width

Twin-width of $(X, <_1, <_2)$:

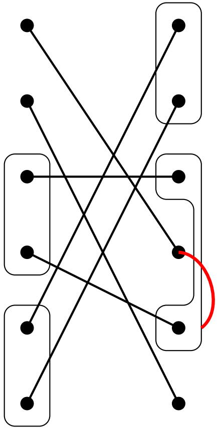
- iteratively merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree



Twin-width

Twin-width of $(X, <_1, <_2)$:

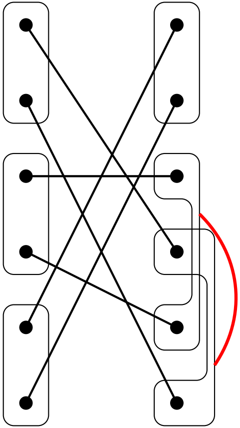
- iteratively merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree



Twin-width

Twin-width of $(X, <_1, <_2)$:

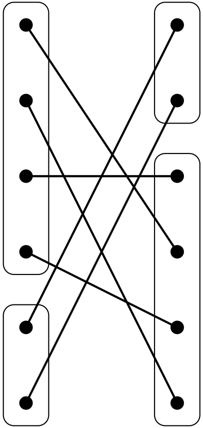
- iteratively merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree



Twin-width

Twin-width of $(X, <_1, <_2)$:

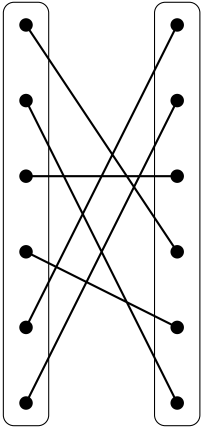
- iterately merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree



Twin-width

Twin-width of $(X, <_1, <_2)$:

- iterratively merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree

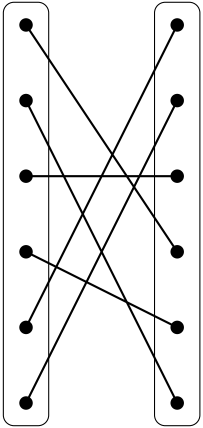


Twin-width

Twin-width of $(X, <_1, <_2)$:

- iteratively merge elements of X
- error between $A, B \subset X$ if they interleave for either $<_1$ or $<_2$
- minimize the error degree

Separable $\iff$ twin-width $= 0$



Guillemot–Marx Algorithm

Theorem (Guillemot–Marx '14)

One can test if τ is a pattern of σ in time $f(\tau) \cdot |\sigma|$.

Win–win argument:

Lemma

A class $\mathcal{C}$ avoids a pattern if and only if it has bounded twin-width.

Lemma

One can test if τ is a pattern of σ in time $f(\tau, \mathrm{tww}(\sigma)) \cdot |\sigma|$.

Pattern-free classes are nice

Theorem (Marcus–Tardos '04)

For any τ, there is a constant c such that $\mathcal{F}(\tau)$ has $\leq c^n$ permutations of size n.

Recognition algorithm:

Theorem (Guillemot–Marx '14)

One can test if τ is a pattern of σ in time $f(\tau) \cdot |\sigma|$.

We give a 'decomposition':

Theorem (BBGT)

For any pattern τ, there is a constant k such that any $\sigma \in \mathcal{F}(\tau)$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with σ_i separable.

Pattern-free classes are nice

Theorem (Marcus–Tardos '04)

For any τ, there is a constant c such that $\mathcal{F}(\tau)$ has $\leq c^n$ permutations of size n.

Recognition algorithm:

Theorem (Guillemot–Marx '14)

One can test if τ is a pattern of σ in time $f(\tau) \cdot |\sigma|$.

We give a 'decomposition':

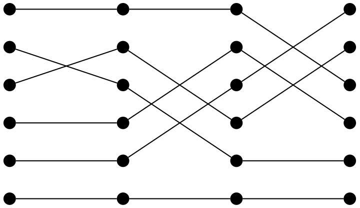
Theorem (BBGT)

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\mathrm{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\mathrm{tww}(\sigma_i) = 0$.

Factorisation

Theorem

For any $t \in \mathbb{N}$, *there is a constant* k *such that any* σ *with* $\mathrm{tww}(\sigma) \leq t$ *factorises as* $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, *with* $\mathrm{tww}(\sigma_i) = 0$.



Factorisation

Theorem

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\operatorname{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\operatorname{tww}(\sigma_i) = 0$.

Fact

For any σ_1, σ_2, $\operatorname{tww}(\sigma_1 \circ \sigma_2) \leq f(\operatorname{tww}(\sigma_1), \operatorname{tww}(\sigma_2))$.

Factorisation

Theorem

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\mathrm{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\mathrm{tww}(\sigma_i) = 0$.

Fact

For any σ_1, σ_2, $\mathrm{tww}(\sigma_1 \circ \sigma_2) \leq f(\mathrm{tww}(\sigma_1), \mathrm{tww}(\sigma_2))$.

Corollary

For a class $\mathcal{C}$ of permutations, TFAE:

- *$\mathcal{C}$ avoids a pattern,*
- *$\mathcal{C}$ has bounded twin-width,*
- *$\mathcal{C} \subset \mathcal{S}^k$ for some $k \in \mathbb{N}$ ($\mathcal{S}$ = separable permutations).*

A class $\mathcal{C}$ of permutations avoids a pattern if and only if $\mathcal{C} \subset \mathcal{S}^k$ for some $k \in \mathbb{N}$.

Proof overview (from very far away)

Theorem

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\mathrm{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\mathrm{tww}(\sigma_i) = 0$.

Main tool for the proof:

Theorem (Pilipczuk & Sokołowski, Bourneuf & Thomassé, '23)

Graphs with bounded twin-width are polynomially χ-bounded.

Proof overview (from very far away)

Theorem

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\mathrm{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\mathrm{tww}(\sigma_i) = 0$.

Main tool for the proof:

Theorem (Pilipczuk & Sokołowski, Bourneuf & Thomassé, '23)

Graphs with bounded twin-width are polynomially χ-bounded.

Behind this theorem:
decomposition of graphs of twin-width k into graphs of twin-width $k-1$.

Proof overview (from very far away)

Theorem

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\mathrm{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\mathrm{tww}(\sigma_i) = 0$.

Main tool for the proof:

Theorem (Pilipczuk & Sokołowski, Bourneuf & Thomassé, '23)

Graphs with bounded twin-width are polynomially χ-bounded.

Behind this theorem:
decomposition of ~~graphs~~ things of twin-width k into things of twin-width $k-1$.

Proof overview (from very far away)

Theorem

For any $t \in \mathbb{N}$, there is a constant k such that any σ with $\operatorname{tww}(\sigma) \leq t$ factorises as $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, with $\operatorname{tww}(\sigma_i) = 0$.

Main tool for the proof:

Theorem (Pilipczuk & Sokołowski, Bourneuf & Thomassé, '23)

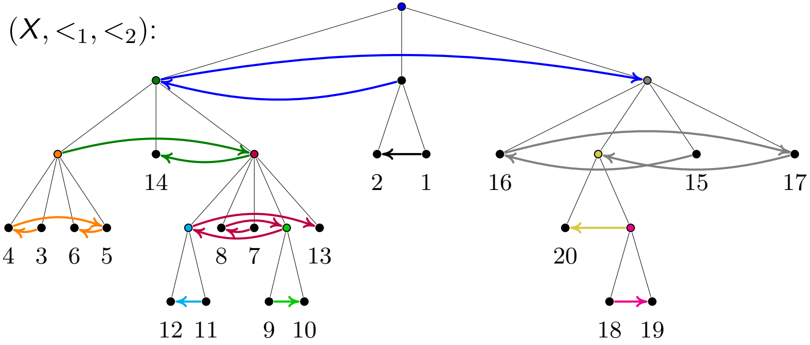
Graphs with bounded twin-width are polynomially χ-bounded.

Behind this theorem:
decomposition of ~~graphs~~ things of twin-width k into things of twin-width $k-1$.

For permutations, this decomposition can be expressed with direct and skew sums, and a bounded number of products.

Substitution

$(X, <_1, <_2)$:

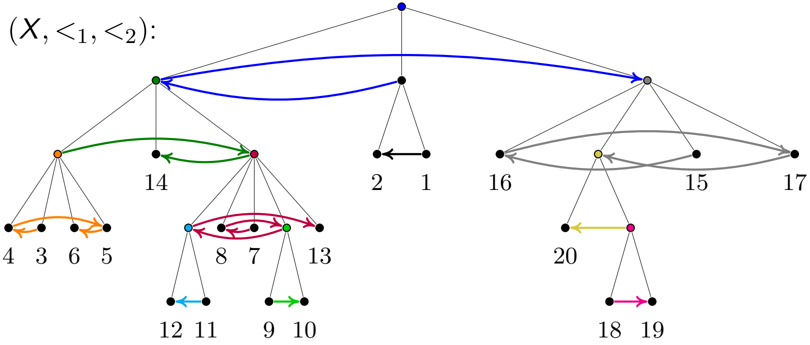


X = leaves, $<_1$ is left-to-right

$x <_2 y$: find the common ancestor t, read the local permutation on the children

Substitution

$(X, <_1, <_2)$:



$X =$ leaves, $<_1$ is left-to-right

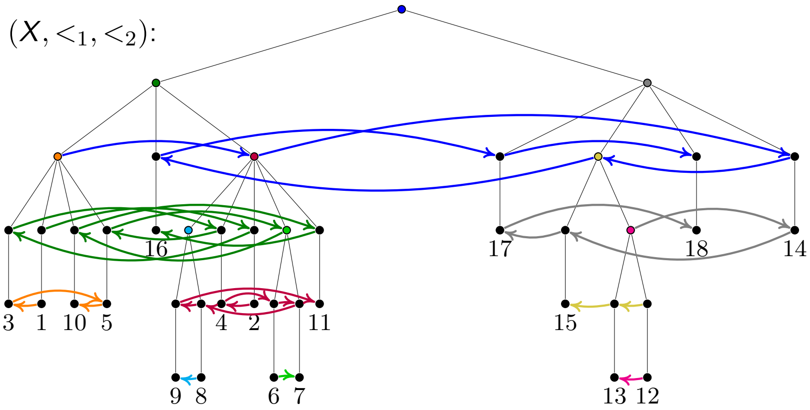
$x <_2 y$: find the common ancestor t, read the local permutation on the children

Lemma

If all local permutations are separable, so is the global permutation.

Delayed Substitution

$(X, <_1, <_2)$:



X = leaves, $<_1$ is left-to-right

$x <_2 y$: find the common ancestor t, read the local permutation on the grandchildren

Using Delayed Substitutions

Key facts:

- [PS, BT] Delayed substitutions can be computed greedily.

Using Delayed Substitutions

Key facts:

- [PS, BT] Delayed substitutions can be computed greedily.
- [PS, BT] In a delayed substitutions, the local permutations are simpler (in the sense of twin-width) than the global permutation.

Using Delayed Substitutions

Key facts:

- [PS, BT] Delayed substitutions can be computed greedily.
- [PS, BT] In a delayed substitutions, the local permutations are simpler (in the sense of twin-width) than the global permutation.
- [BBGT] Any delayed substitution of permutations in $\mathcal{C}$ can be rewritten into 2 compositions of substitutions of permutations in $\mathcal{C}$.

Using Delayed Substitutions

Key facts:

- [PS, BT] Delayed substitutions can be computed greedily.
- [PS, BT] In a delayed substitutions, the local permutations are simpler (in the sense of twin-width) than the global permutation.
- [BBGT] Any delayed substitution of permutations in $\mathcal{C}$ can be rewritten into 2 compositions of substitutions of permutations in $\mathcal{C}$.

Using Delayed Substitutions

Key facts:

- [PS, BT] Delayed substitutions can be computed greedily.
- [PS, BT] In a delayed substitutions, the local permutations are simpler (in the sense of twin-width) than the global permutation.
- [BBGT] Any delayed substitution of permutations in $\mathcal{C}$ can be rewritten into 2 compositions of substitutions of permutations in $\mathcal{C}$.

To factorise a permutation σ:

- compute the delayed substitution for σ,
- recursively factorise the local permutations,
- rewrite into composition of substitution of separable permutations (using some distributive property)

Application to Graphs

Corollary (Sparse graphs)

There are $f : \mathbb{N} \to \mathbb{N}$ and $c \in \mathbb{N}$ satisfying the following: if G has no $K_{t,t}$-subgraph and $\mathrm{tww}(G) \leq k$, then the $f(k,t)$-subdivision of G has twin-width $\leq c$.

Application to Graphs

Corollary (Sparse graphs)

There are $f : \mathbb{N} \to \mathbb{N}$ *and* $c \in \mathbb{N}$ *satisfying the following: if* G *has no* $K_{t,t}$*-subgraph and* $\mathrm{tww}(G) \leq k$*, then the* $f(k,t)$*-subdivision of* G *has twin-width* $\leq c$*.*

Corollary (General case)

There is a constant c *such that for any* k*, there is an encoding of graphs of twin-width* k *into graphs of twin-width* c *expressed with first-order logic.*

Application to Graphs

Corollary (Sparse graphs)

There are $f : \mathbb{N} \to \mathbb{N}$ *and* $c \in \mathbb{N}$ *satisfying the following: if* G *has no* $K_{t,t}$*-subgraph and* $\mathrm{tww}(G) \leq k$*, then the* $f(k,t)$*-subdivision of* G *has twin-width* $\leq c$*.*

Corollary (General case)

There is a constant c *such that for any* k*, there is an encoding of graphs of twin-width* k *into graphs of twin-width* c *expressed with first-order logic.*

Conjecture: $c = 4$ can be reached for both results.

Open Questions

- Algorithmic applications of the factorisation?
- Computing shortest factorisations into separable permutations? (is it FPT? approximation?)
- Generalisation to matrices?

Open Questions

- Algorithmic applications of the factorisation?
- Computing shortest factorisations into separable permutations? (is it FPT? approximation?)
- Generalisation to matrices?

Thank you!