Meta-Kernelization with Structural Parameters

Robert Ganian Friedrich Slivovsky Stefan Szeider

Goethe University Frankfurt, Germany
Vienna University of Technology, Austria

Worker 2013, Warsaw

Motivation

The aim of meta-kernelization is to obtain polykernels for *large classes of problems*.

Motivation

The aim of meta-kernelization is to obtain polykernels for *large classes of problems*.
Several interesting results – see e.g.:

1. Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos:

Theorem

Let $\mathcal{P} \subseteq G \times \mathbb{N}$ have finite integer index and let either $\mathcal{P}$ or $\bar{\mathcal{P}}$ be quasi-compact. Then $\mathcal{P}$ admits a linear kernel.

2. Fomin, Lokshtanov, Misra, Saurabh:

Theorem

For every set $\mathcal{F} \in \mathbb{F}$, p-$\mathcal{F}$-Deletion admits a polynomial kernel.

Motivation

The aim of meta-kernelization is to obtain polykernels for *large classes of problems*.
Several interesting results – see e.g.:

1. Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos:

Theorem

Let $\mathcal{P} \subseteq G \times \mathbb{N}$ have finite integer index and let either $\mathcal{P}$ or $\bar{\mathcal{P}}$ be quasi-compact. Then $\mathcal{P}$ admits a linear kernel.

2. Fomin, Lokshtanov, Misra, Saurabh:

Theorem

For every set $\mathcal{F} \in \mathbb{F}$, p-$\mathcal{F}$-Deletion admits a polynomial kernel.

- Both of these examples use the *solution size* as the parameter.
- But what if the solution size is large? Can we use *structural parameters*?

Motivation II

- Strong evidence that many FPT graph problems parameterized by tree-width or clique-width are **highly unlikely to admit polykernels**.
- Use of weaker structural parameters for individual problems: vertex cover number, max-leaf number, neighborhood diversity...
- Little known about meta-kernelization with structural parameters (until recently)

Our results

Let $\mathcal{C}$ be a graph class of bounded rank-width (or equivalently bounded clique-width or bounded boolean width).

Our results

Let $\mathcal{C}$ be a graph class of bounded rank-width (or equivalently bounded clique-width or bounded boolean width).

Theorem

Every **MSO model checking problem**, *parameterized by the* $\mathcal{C}$-**cover number** *of the input graph, has a polynomial kernel with a linear number of vertices.*

- Polykernels for c-Coloring, c-Domatic number, Independent Dominating Set and many other problems.

Our results

Let $\mathcal{C}$ be a graph class of bounded rank-width (or equivalently bounded clique-width or bounded boolean width).

Theorem

Every **MSO model checking problem**, *parameterized by the* $\mathcal{C}$-**cover number** *of the input graph, has a polynomial kernel with a linear number of vertices.*

- Polykernels for c-Coloring, c-Domatic number, Independent Dominating Set and many other problems.

Theorem

Every **MSO optimization problem**, *parameterized by the* $\mathcal{C}$-**cover number** *of the input graph, has a polynomial bikernel with a linear number of vertices.*

- Polykernels for Dominating Set, (Connected) Vertex Cover, Feedback Vertex Set and many others, even if the solution size is huge.

MSO (MS_1) logic

The language

- vertex variables $x, y, z, \ldots$
- vertex set variables $X, Y, Z, \ldots$
- logic connectives $\vee, \wedge, \rightarrow, \ldots$
- quantification over (sets of) vertices $\forall v \in X, \exists Z, \ldots$
- edge predicate edge(x, y)
- *no* quantification over *edges* in MS_1

MSO (MS_1) logic

The language

- vertex variables $x, y, z, \ldots$
- vertex set variables $X, Y, Z, \ldots$
- logic connectives $\vee, \wedge, \rightarrow, \ldots$
- quantification over (sets of) vertices $\forall v \in X, \exists Z, \ldots$
- edge predicate edge(x, y)
- *no* quantification over *edges* in MS_1

Expressible properties

- 3-colorability, independent dominating set, ...
- $\alpha \equiv \forall X \, \exists y \in X \, \exists z \notin X : \text{edge}(z, y)$ (connectivity)

MSO (MS_1) logic

The language

- vertex variables $x, y, z, \ldots$
- vertex set variables $X, Y, Z, \ldots$
- logic connectives $\vee, \wedge, \rightarrow, \ldots$
- quantification over (sets of) vertices $\forall v \in X, \exists Z, \ldots$
- edge predicate edge(x, y)
- *no* quantification over *edges* in MS_1

Expressible properties

- 3-colorability, independent dominating set, ...
- $\alpha \equiv \forall X \, \exists y \in X \, \exists z \notin X : \text{edge}(z, y)$ (connectivity)

Definition (MSO-MC_ϕ)

Instance: A graph G.
Question: Does $G \models \phi$ hold?

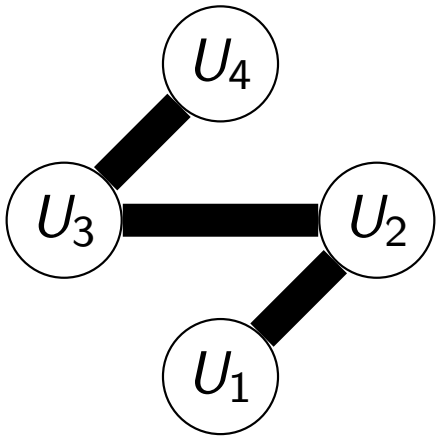
Sketching the parameter: $\mathcal{C}$-covers

- For a graph class $\mathcal{C}$, let a *$\mathcal{C}$-cover of a graph G* be a partition of the vertex set into modules* $\{U_1, \ldots, U_k\}$ where each module induces a subgraph which belongs to $\mathcal{C}$.

* *All vertices in U_i have the same neighborhood outside of U_i.*

Sketching the parameter: $\mathcal{C}$-covers

- For a graph class $\mathcal{C}$, let a *$\mathcal{C}$-cover of a graph G* be a partition of the vertex set into modules* $\{U_1, \ldots, U_k\}$ where each module induces a subgraph which belongs to $\mathcal{C}$.



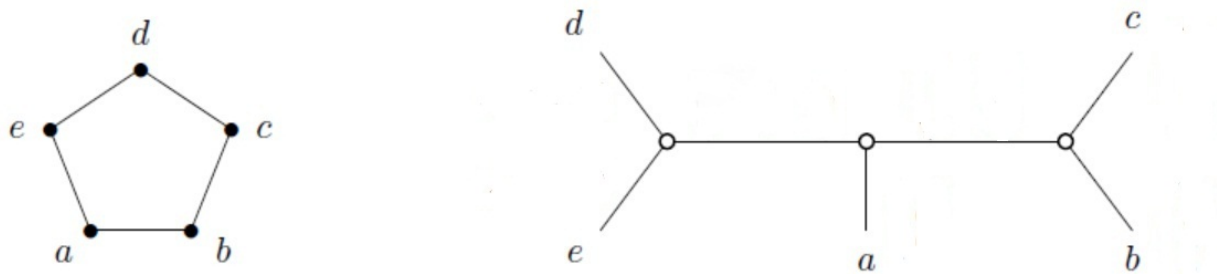
- The *$\mathcal{C}$-cover number* is then the size of a smallest $\mathcal{C}$-cover of G.

* *All vertices in U_i have the same neighborhood outside of U_i.*

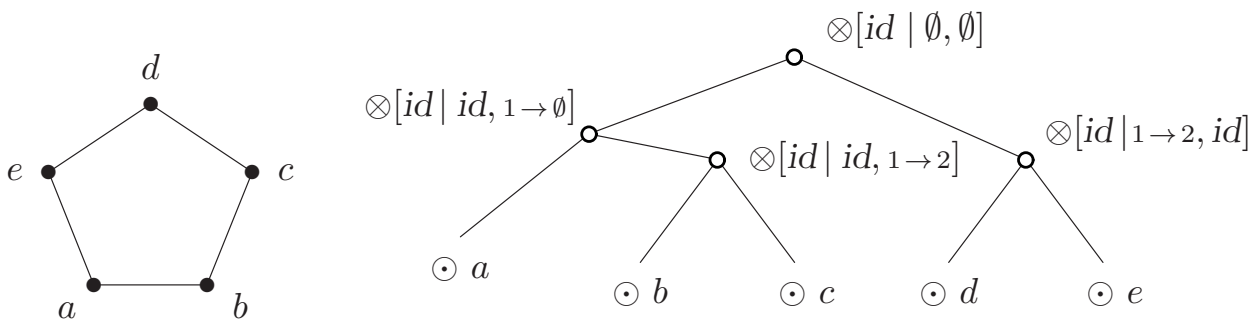
Rank-width covers

Rank-width:

- Definition fairly technical + we won't need it much



A rank-decomposition of C_5 (width 2).



A 2-labeling parse tree of C_5.

Rank-width covers

Rank-width:

- Definition fairly technical.
- Related to clique-width, but may be computed **in FPT time**.
- Can be used to solve MSO model-checking in FPT time – more about this later.

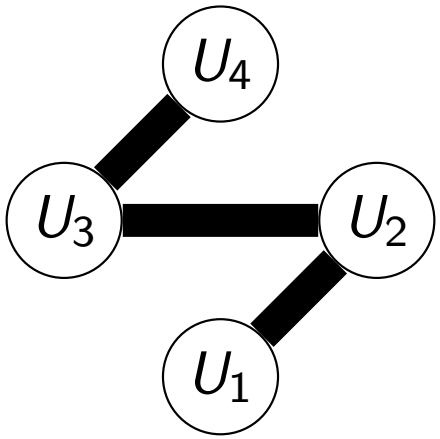
Rank-width covers

Rank-width:

- Definition fairly technical.
- Related to clique-width, but may be computed **in FPT time**.
- Can be used to solve MSO model-checking in FPT time – more about this later.

Definition

A *rank-width-d cover* of a graph G is a $\mathcal{C}$-cover of G where $\mathcal{C}$ is the class of graphs of rank-width at most d.



Rank-width covers

Rank-width:

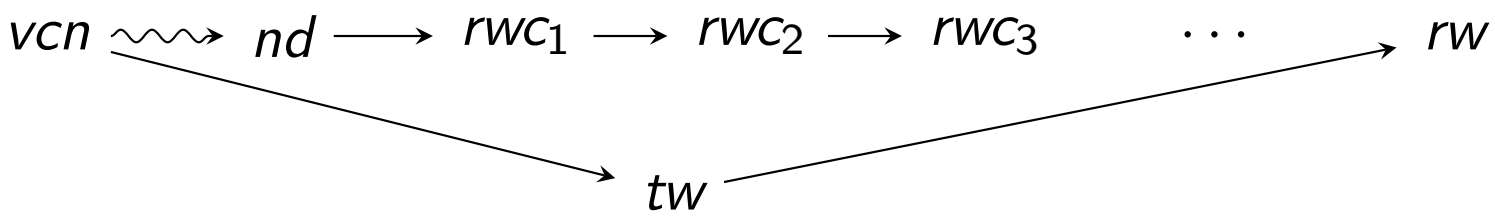
- Definition fairly technical.
- Related to clique-width, but may be computed **in FPT time**.
- Can be used to solve MSO model-checking in FPT time – more about this later.

Definition

A *rank-width-d cover* of a graph G is a $\mathcal{C}$-cover of G where $\mathcal{C}$ is the class of graphs of rank-width at most d.

- d is a constant in our setting (we can have rank-width-1 covers, rank-width-2 covers etc.)
- The value of d allows us to **scale** the parameter to our needs: a larger d may reduce the size of our kernels at the cost of higher constants in the runtime.

Rank-width covers: where do they fit?



Relationship between graph invariants:

- vertex cover number (vcn),
- neigborhood diversity (nd),
- rank-width-d cover number (rwc_d),
- rank-width (rw),
- and treewidth (tw).

$A \to B$ indicates that any graph class where A is bounded also has bounded B.

Finding rank-width covers

Theorem

A smallest rank-width-d cover of a graph can be computed in **polynomial time**.

- Not possible for any of the popular parameters (FPT at best).
- Otherwise we would need to receive a rank-width-d cover from an oracle (as is the case with clique-width and clique-decompositions).
- Note: The smallest rank-width-d cover is unique for each d.

Finding rank-width covers: Proof overview

Definition

For two vertices $a, b \in V(G)$, let $a \sim_d b$ iff there is a module M in G containing a, b such that $rw(G[M]) \leq d$.

Proposition

$\sim_d$ is an equivalence relation, and each equivalence class of $\sim_d$ is a module of G with rank-width at most d.

Finding rank-width covers: Proof overview

Definition

For two vertices $a, b \in V(G)$, let $a \sim_d b$ iff there is a module M in G containing a, b such that $rw(G[M]) \leq d$.

Proposition

$\sim_d$ is an equivalence relation, and each equivalence class of $\sim_d$ is a module of G with rank-width at most d.

- Reflexivity and symmetry – immediate.
- For the rest, we use a technical lemma which says that if two modules with $rw \leq d$ intersect, then their union is a module with $rw \leq d$.

Finding rank-width covers: Proof overview

Definition

For two vertices $a, b \in V(G)$, let $a \sim_d b$ iff there is a module M in G containing a, b such that $rw(G[M]) \leq d$.

Proposition

$\sim_d$ is an equivalence relation, and each equivalence class of $\sim_d$ is a module of G with rank-width at most d.

- Reflexivity and symmetry – immediate.
- For the rest, we use a technical lemma which says that if two modules with $rw \leq d$ intersect, then their union is a module with $rw \leq d$.
- How to get transitivity: $a \sim_d b$ and $b \sim_d c$ implies the existence of two modules with $rw \leq d$ intersecting in b.

Finding rank-width covers: Proof overview II

Proposition

$\sim_d$ is an equivalence relation, and each equivalence class of $\sim_d$ is a module of G with rank-width at most d.

Corollary: the equivalence classes of $\sim_d$ form a smallest rank-width-d cover of G.

Finding rank-width covers: Proof overview II

Proposition

$\sim_d$ is an equivalence relation, and each equivalence class of $\sim_d$ is a module of G with rank-width at most d.

Corollary: the equivalence classes of $\sim_d$ form a smallest rank-width-d cover of G.

Proposition

Let $d \in \mathbb{N}$ be a constant. Given a graph G and two vertices $v, w \in V(G)$, we can decide whether $v \sim_d w$ in polynomial time.

- We use modular decompositions to compute the unique inclusion-minimal module M containing v, w in quadratic time*.
- Rank-width is closed under induced subgraphs, so $v \sim_d w$ iff $rw(G[M]) \leq d$, which can be checked in cubic time.

* *Bui-Xuan, Habib, Limouzy and Montgolfier, DAM 2009.*

Intermezzo: MSO model checking on rank-width

A t-labeled graph $\bar{G}$ is a graph together with a labeling $lab : V \to 2^{\{1,\ldots,t\}}$. For now, assume $t = 1$; then the join operator $\bar{G} \otimes \bar{H}$ does the following:

1. makes a disjoint union of the graphs,
2. adds edges between $v \in G, w \in H$ which both had the label 1.

Note: rank-width may be defined as the "minimum number of labels necessary to construct the graph using $\otimes$ and relabeling".

Intermezzo: MSO model checking on rank-width

A t-labeled graph $\bar{G}$ is a graph together with a labeling $lab : V \to 2^{\{1,\ldots,t\}}$. For now, assume $t = 1$; then the join operator $\bar{G} \otimes \bar{H}$ does the following:

1. makes a disjoint union of the graphs,
2. adds edges between $v \in G, w \in H$ which both had the label 1.

Note: rank-width may be defined as the "minimum number of labels necessary to construct the graph using $\otimes$ and relabeling".

Definition (Canonical equivalence)

Let ϕ be an MSO formula. The t- labeled graphs $\bar{G}_1$ and $\bar{G}_2$ are *canonically equivalent*, in symbols $\bar{G}_1 \approx_{\phi,t} \bar{G}_2$, if for all t-labeled graphs $\bar{H}$ we have $(\bar{G}_1 \otimes \bar{H}) \models \phi \Leftrightarrow (\bar{G}_2 \otimes \bar{H}) \models \phi$.

Intermezzo: MSO model checking on rank-width

Definition (Canonical equivalence)

Let ϕ be an MSO formula. The t- labeled graphs $\bar{G}_1$ and $\bar{G}_2$ are *canonically equivalent*, in symbols $\bar{G}_1 \approx_{\phi,t} \bar{G}_2$, if for all t-labeled graphs $\bar{H}$ we have $(\bar{G}_1 \otimes \bar{H}) \models \phi \Leftrightarrow (\bar{G}_2 \otimes \bar{H}) \models \phi$.

Theorem (MSO-MC on rank-width*)

Let $t \geq 1$ be fixed, and ϕ fixed MSO formula. Then the following holds.

1. *$\approx_{\phi,t}$ has finite index in the universe of t-labeled graphs.*
2. *For any t-labeled input graph $\bar{G}$ of rank-width at most t, it is possible to compute the equiv. class $[\bar{G}]$ w.r.t. $\approx_{\phi,t}$ in polynomial time through a leaves-to-root tree automaton.*

* *Ganian and Hliněný, DAM 2010.*

Obtaining small representatives

Recall that each U_i has rank-width at most d (a constant).
Let $\bar{U}_i$ be the graph U_i with every vertex labeled by $\{1\}$.

Obtaining small representatives

Recall that each U_i has rank-width at most d (a constant).
Let $\bar{U}_i$ be the graph U_i with every vertex labeled by $\{1\}$.
Then

Theorem (MSO-MC on rank-width)

Let $t \geq 1$ be fixed, and ϕ fixed MSO formula. Then the following holds.

3. *For any t-labeled input graph $\bar{G}$ of rank-width at most t, it is possible to compute $[\bar{G}]$ with respect to $\approx_{\phi,t}$ in polynomial time through a leaves-to-root tree automaton.*

allows us to compute $[\bar{U}_i]$ in polytime.

Obtaining small representatives

Furthermore,

Theorem (MSO-MC on rank-width)

Let $t \geq 1$ be fixed, and ϕ fixed MSO formula. Then the following holds.

1. *$\approx_{\phi,t}$ has finite index in the universe of t-labeled graphs.*

means that we can have a constant-size fully-labeled representative for each equivalence class of $\approx_{\phi,t}$.

Obtaining small representatives

Furthermore,

Theorem (MSO-MC on rank-width)

Let $t \geq 1$ be fixed, and ϕ fixed MSO formula. Then the following holds.

1. *$\approx_{\phi,t}$ has finite index in the universe of t-labeled graphs.*

means that we can have a constant-size fully-labeled representative for each equivalence class of $\approx_{\phi,t}$.
Intuitive idea:

- The number of equivalence classes does not depend on the input, just on d and ϕ.
- Each equivalence class has some representative of minimum size, and this also depends only on d and ϕ.
- Brute force can be used to iteratively process all graphs with $1, 2, \ldots$ vertices until a representative is found for each equivalence class.

Kernels for MSO model checking

Theorem

Every MSO model checking problem, parameterized by the rank-width-d cover number of the input graph, admits a linear kernel.

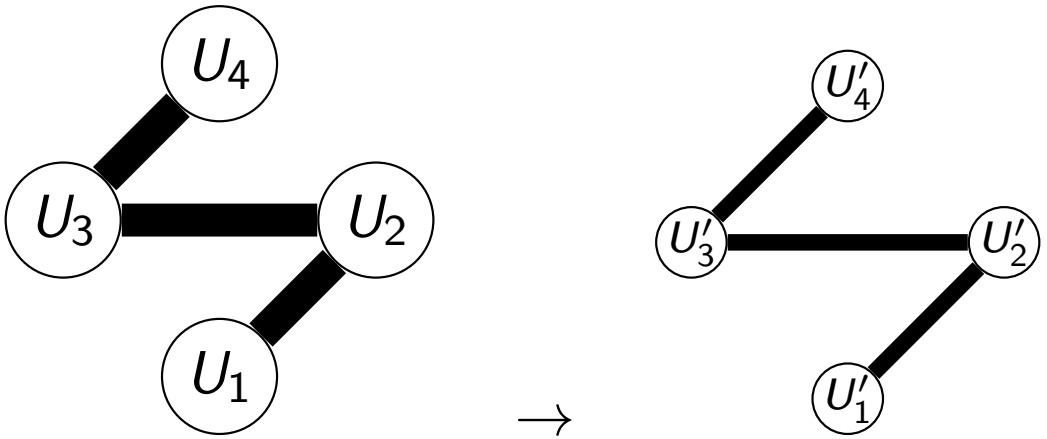
Let G be the input graph. To obtain the kernel:

1. compute the rank-width-d cover $\{U_1, \ldots, U_k\}$ in polytime,
2. compute each equivalence class $[\bar{U}_i]$ in polytime,
3. obtain G' from G by replacing each U_i by the representative of $[\bar{U}_i]$,
4. output G' (without the labels).

Kernels for MSO model checking

Theorem

Every MSO model checking problem, parameterized by the rank-width-d cover number of the input graph, admits a linear kernel.



The size of each U'_i is constant, and their number is k.

Why does it work?

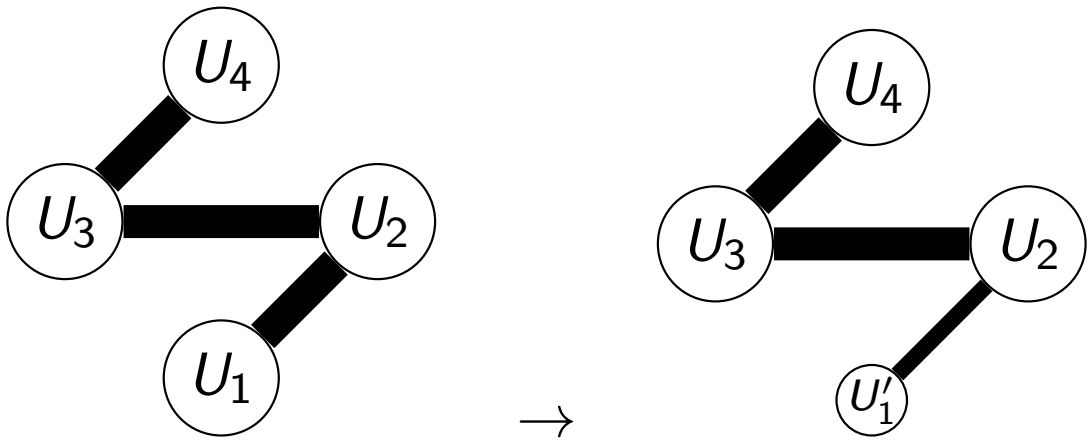
Let $G_0 = G$ and G_1 be the graph obtained by replacing U_1 by U_1'. Since $\bar{U}_1 \approx_{\phi,t} \bar{U}_1'$, we have that

$$\bar{U}_1 \otimes \bar{H} \models \phi \quad \textit{iff} \quad \bar{U}_1' \otimes \bar{H} \models \phi$$

We may choose $\bar{H}$ to be the 1-labeling of $\bar{G} - U_1$ which produces G_0 and G_1 from $\bar{U}_1 \otimes \bar{H}$ and $\bar{U}_1' \otimes \bar{H}$ respectively, and thus

$$G_0 \models \phi \quad \textit{iff} \quad G_1 \models \phi$$

.



MSO optimization problems

- MSO formulas can only directly capture decision problems such as 3-colorability.
- Several authors* have extended the expressive power of MSO logic to also capture optimization problems.

** Arnborg, Lagergren and Seese, and later Courcelle, Makowsky and Rotics.*

MSO optimization problems

- MSO formulas can only directly capture decision problems such as 3-colorability.
- Several authors* have extended the expressive power of MSO logic to also capture optimization problems.

Let ϕ be a fixed MSO formula with one free set variable and $\diamondsuit \in \{\geq, \leq\}$. Then:

Definition ($\text{MSO-OPT}_{\phi}^{\diamondsuit}$)

Instance: A graph G and an integer $r \in \mathbb{N}$.
Question: Is there a set $A \subseteq V(G)$ such that $G \models \phi(A)$ and $|A| \diamondsuit r$.

* *Arnborg, Lagergren and Seese, and later Courcelle, Makowsky and Rotics.*

MSO optimization problems

- MSO formulas can only directly capture decision problems such as 3-colorability.
- Several authors* have extended the expressive power of MSO logic to also capture optimization problems.

Let ϕ be a fixed MSO formula with one free set variable and $\diamondsuit \in \{\geq, \leq\}$. Then:

Definition ($\text{MSO-OPT}_{\phi}^{\diamondsuit}$)

Instance: A graph G and an integer $r \in \mathbb{N}$.
Question: Is there a set $A \subseteq V(G)$ such that $G \models \phi(A)$ and $|A| \diamondsuit r$.

Our results may be straightforwardly extended to minimize/maximize linear expressions such as $2|A| - 9.5|B| \ldots$.

* *Arnborg, Lagergren and Seese, and later Courcelle, Makowsky and Rotics.*

Kernels for MSO optimization problems

Can we use the same approach to get a polykernel for $\text{MSO-OPT}_{\phi}^{\diamondsuit}$?

Kernels for MSO optimization problems

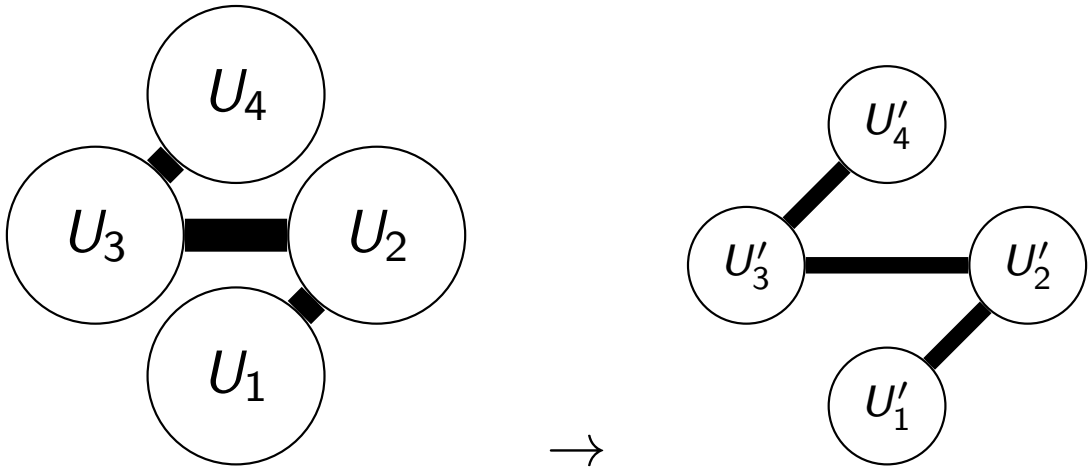
Can we use the same approach to get a polykernel for $\text{MSO-OPT}_{\phi}^{\diamondsuit}$?

Obstacle no. 1: Instead of simply deciding $G \models \phi$, we now need to deal with a graph together with some specific vertex-subset A (we call such graphs *equipped*).

- Luckily, the "MSO-MC on rank-width" theorem extends nicely from closed formulas to formulas with free variables.
- The canonical equivalence for $\phi(A)$ has finite index on all equipped labeled graphs (of bounded rank-width).

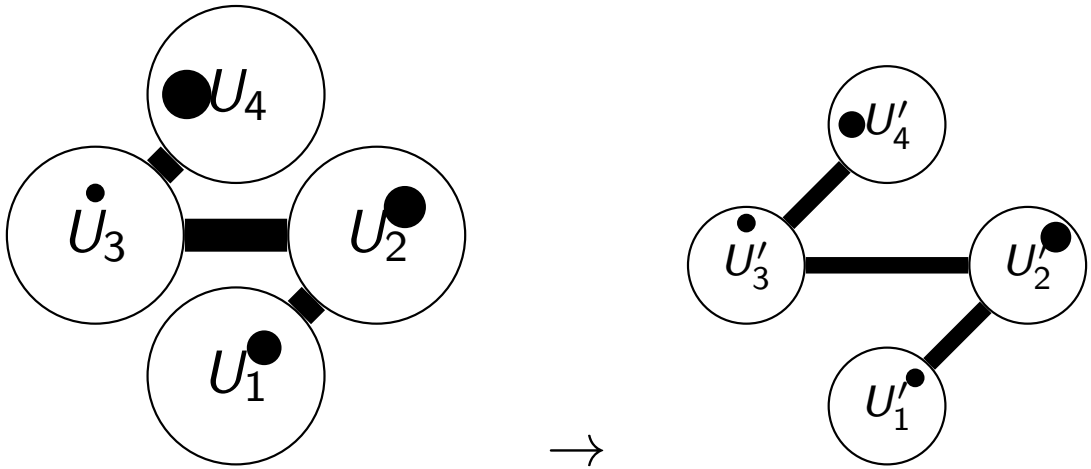
Kernels for MSO optimization problems – Naive idea

This allows us to replace each U_i by a constant-size representative U'_i such that they are equivalent for *all possible equipments of A*.



Kernels for MSO optimization problems – Naive idea

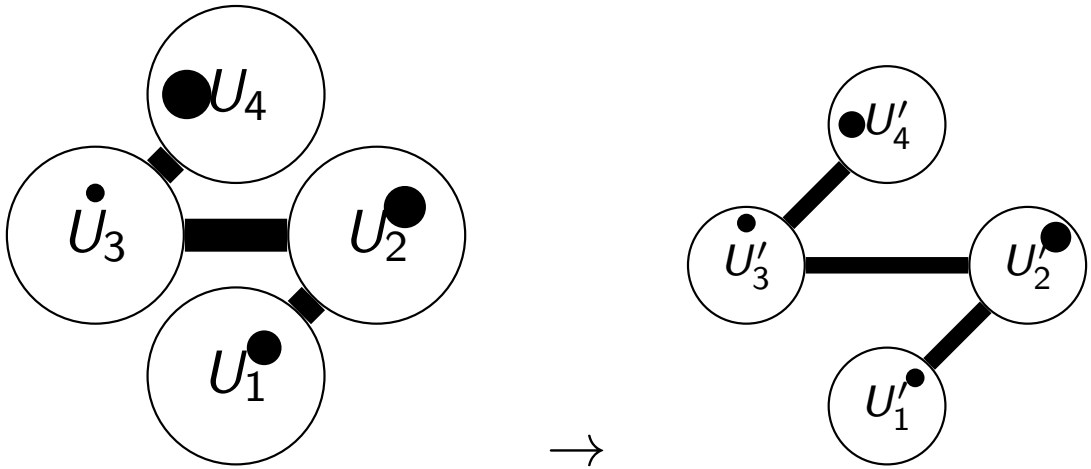
This allows us to replace each U_i by a constant-size representative U_i' such that they are equivalent for *all possible equipments of A*.



For each A in G we would have an A' in G' such that $G \models \phi(A)$ *iff* $G' \models \phi(A')$.

Kernels for MSO optimization problems – Naive idea

This allows us to replace each U_i by a constant-size representative U'_i such that they are equivalent for *all possible equipments of A*.



For each A in G we would have an A' in G' such that $G \models \phi(A)$ *iff* $G' \models \phi(A')$.

But how does this help us decide whether $|A| \diamondsuit r$?

Definition ($\text{MSO-OPT}^{\diamondsuit}_{\phi}$)

Instance: A graph G and an integer $r \in \mathbb{N}$.

Question: Is there a set $A \subseteq V(G)$ such that $G \models \phi(A)$ and $|A| \diamondsuit r$.

Kernels for MSO optimization problems

Obstacle no. 2: When replacing the modules U_i by their smaller representatives U_i', we lose track of the original size of A.

Kernels for MSO optimization problems

Obstacle no. 2: When replacing the modules U_i by their smaller representatives U_i', we lose track of the original size of A.

- This can be solved by storing the optimal original size of A which corresponds to each equipment A' in each U_i'.
- Since the size of each U_i' is constant, we only need to store $O(k)$ values. Instead of a kernel we get an *annotated kernel* – a special case of a bikernel.

Kernels for MSO optimization problems

Obstacle no. 2: When replacing the modules U_i by their smaller representatives U_i', we lose track of the original size of A.

- This can be solved by storing the optimal original size of A which corresponds to each equipment A' in each U_i'.
- Since the size of each U_i' is constant, we only need to store $O(k)$ values. Instead of a kernel we get an *annotated kernel* – a special case of a bikernel.

Obstacle no. 3: The values we store can be large ($\log n$ bits) – much larger than the $poly(k)$ size bound for a kernel.

Kernels for MSO optimization problems

Obstacle no. 2: When replacing the modules U_i by their smaller representatives U_i', we lose track of the original size of A.

- This can be solved by storing the optimal original size of A which corresponds to each equipment A' in each U_i'.
- Since the size of each U_i' is constant, we only need to store $O(k)$ values. Instead of a kernel we get an *annotated kernel* – a special case of a bikernel.

Obstacle no. 3: The values we store can be large ($\log n$ bits) – much larger than the $poly(k)$ size bound for a kernel.

- We solve this final obstacle by comparing 2^k and n.
- If $k \geq \log n$, then the $\log n$ bits necessary to store our annotation still "fit" inside our annotated kernel.
- If $k < \log n$, then we use brute force in our "kernel" to solve the problem outright in polytime (and then replace it with a trivial yes/no instance for the kernel).

Final notes

Why choose rank-width?

- it is the most general known parameter which can be computed efficiently (in FPT time) and handles MSO.

Can these results be extended to MSO_2 with tree-width?

- Not directly, due to the unbounded number of edges between modules.
- If we destroy the modular structure, then it is not clear how to compute the cover.

Thank you for your attention.