

Parameterized algorithms for THINNESS via the cluster module number

Flavia Bonomo, Eric Brandwein, Ignasi Sau

October 17, 2024

Table of Contents

- 1 Thinness
- 2 Parameterized complexity
- 3 Reducing thinness instances
- 4 Cluster module number
- 5 Parameterizations

Thickness

Consistent solution

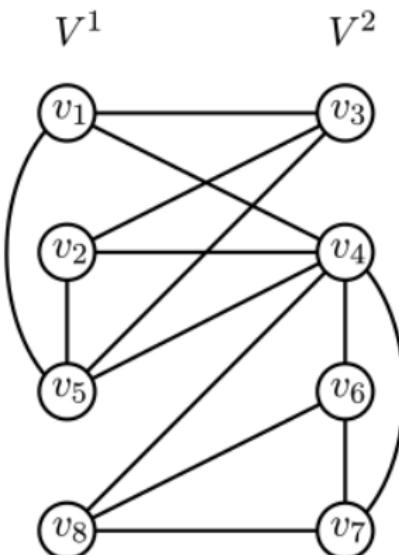
A *consistent solution using k classes* for a graph G is a pair (\prec, S) where

- \prec is a strict total order on $V(G)$, and
- S is a partition of $V(G)$ into k classes,

such that, for each triple of vertices $u \prec v \prec w$ of G , if

- u and v belong to the same class in S , and
- $(u, w) \in E(G)$,

then $(v, w) \in E(G)$.



Definition

A graph is *k-thin* if it admits a consistent solution using *k* classes.
The *thinness* of a graph G , denoted by $\text{thin}(G)$, is the minimum integer k such that G admits a consistent solution using k classes.

THINNESS problem

Input: A graph G and an integer k .

Output: Is $\text{thin}(G) \leq k$?

Thinness

Theorem (Y. Shitov, [5])

THINNESS is NP-complete.

Thinness

Theorem (Y. Shitov, [5])

THINNESS is NP-complete.

Question: What are some classes of graphs for which THINNESS has a polynomial-time algorithm?

Thinness

Theorem (Y. Shitov, [5])

THINNESS is NP-complete.

Question: What are some classes of graphs for which THINNESS has a polynomial-time algorithm?

We know for example that trees admit an $O(n \log n)$ algorithm for THINNESS [1].

Thinness

Theorem (Y. Shitov, [5])

THINNESS is NP-complete.

Question: What are some classes of graphs for which THINNESS has a polynomial-time algorithm?

We know for example that trees admit an $O(n \log n)$ algorithm for THINNESS [1].

Maybe there is some number we can calculate for a graph G that can tell us how hard it is to solve THINNESS for G ?

Parameterized complexity

Classical problems

- Input: String of bits
- Output: Yes or No
- Time complexity measured on the **size of the input.**

Parameterized problems

- Input: String of bits + **parameter k (an integer)**
- Output: Yes or No
- Time complexity measured on the **size of the input and the value of k .**

FPT complexity class

A parameterized problem is in the class FPT (Fixed Parameter Tractable) if there is an algorithm that solves it in time $O(f(k) \cdot n^c)$, where f is a computable function and c is a constant.

FPT complexity class

A parameterized problem is in the class FPT (Fixed Parameter Tractable) if there is an algorithm that solves it in time $O(f(k) \cdot n^c)$, where f is a computable function and c is a constant.

Examples:

- k -VERTEX COVER.

FPT complexity class

A parameterized problem is in the class FPT (Fixed Parameter Tractable) if there is an algorithm that solves it in time $O(f(k) \cdot n^c)$, where f is a computable function and c is a constant.

Examples:

- k -VERTEX COVER.
- Many problems parameterized by the treewidth of the input graph.

FPT complexity class

A parameterized problem is in the class FPT (Fixed Parameter Tractable) if there is an algorithm that solves it in time $O(f(k) \cdot n^c)$, where f is a computable function and c is a constant.

Examples:

- k -VERTEX COVER.
- Many problems parameterized by the treewidth of the input graph.
- THINNESS parameterized by the cluster module number, neighborhood diversity, twin-cover, or vertex cover of the input graph (Our work).

Kernelization

A *kernel* for a parameterized problem is a polynomial-time algorithm that, given an instance (I, k) for the problem, outputs an *equivalent* instance (I', k') such that:

- $|I'| + k' \leq g(k)$ for some computable function g , and
- (I, k) is a YES instance iff (I', k') is a YES instance.

Kernelization

Theorem ([3])

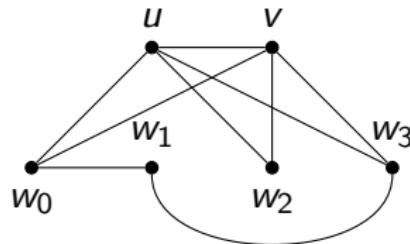
A parameterized problem is in FPT iff it admits a kernel.

Reducing thinness instances

Reducing thinness instances

Lemma 1 ([2])

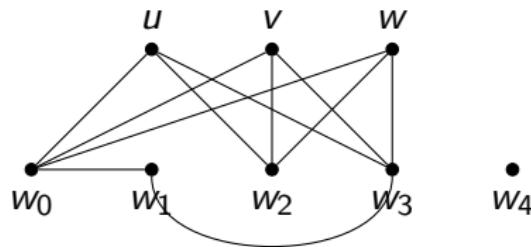
If two adjacent vertices have the same neighborhood, removing one of them does not modify the thinness of the graph.



Reducing thinness instances

Lemma 2

If three **non-adjacent** vertices have the same neighborhood, removing one of them does not modify the thinness of the graph.



Reducing thinness instances

What if the graph had few vertices after applying Lemmas 1 and 2 exhaustively?

Reducing thinness instances

What if the graph had few vertices after applying Lemmas 1 and 2 exhaustively?

Say, at most $g(k)$ vertices for some k ?

Reducing thinness instances

What if the graph had few vertices after applying Lemmas 1 and 2 exhaustively?

Say, at most $g(k)$ vertices for some k ?

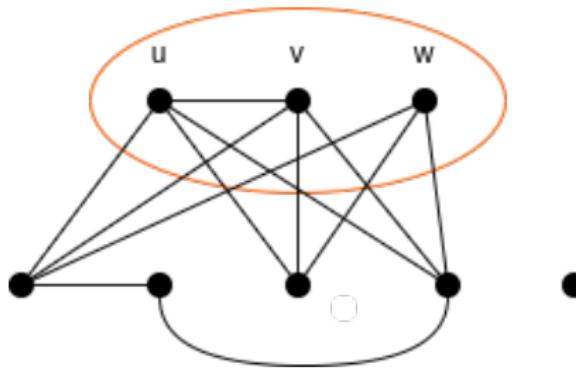
What would that k measure?

Cluster module number

Modules

Definition (Module)

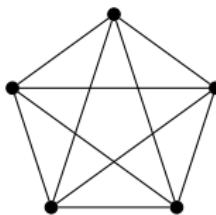
A *module* of a graph G is a subset W of $V(G)$ such that every vertex outside W is either adjacent to all vertices in W or to none of them.



Cluster modules

Definition (Clique)

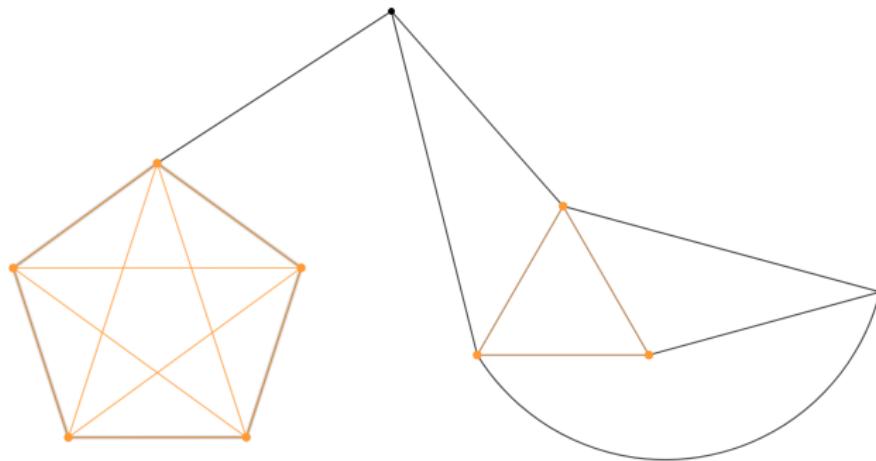
A *clique* is a subset C of vertices such that every vertex in W is adjacent to all other vertices in W .



Cluster modules

Definition (Cluster)

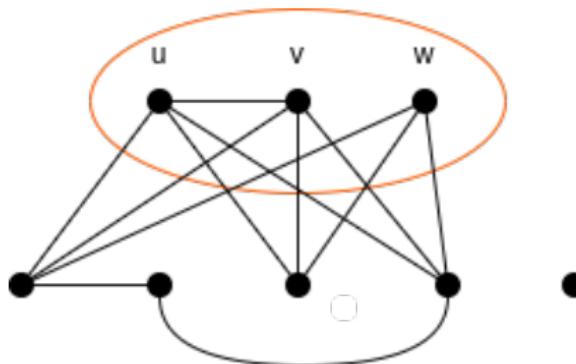
A *cluster* is a subset C of vertices that induces a union of cliques.



Cluster modules

Definition (Cluster module)

A *cluster module* is a subset C of vertices that is a cluster and a module of G .



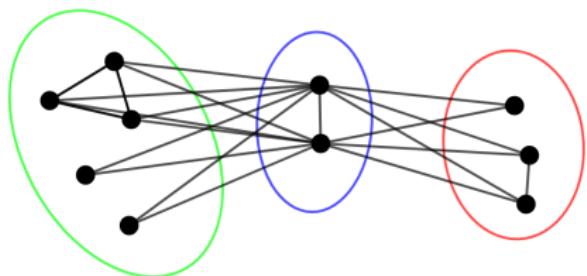
Cluster module number

Definition

A *cluster module partition* of a graph G is a partition of the vertices of G into cluster modules.

Definition

The *cluster module number* $\text{cm}(G)$ of a graph G is the minimum size of a cluster module partition of G .



Cluster module number

Theorem

The cluster module number of a graph can be calculated in linear time.

Parameterizations

Cluster module number

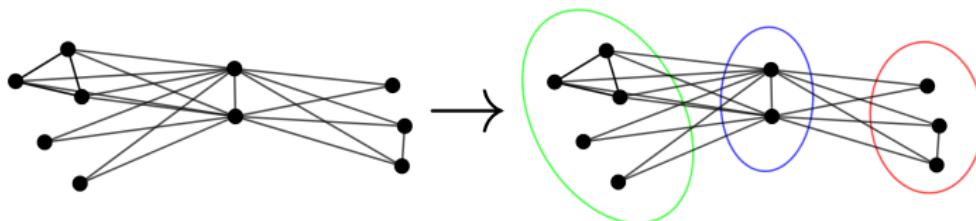
Theorem

THINNESS admits a kernel of size $2 \cdot \text{cm}(G)$ when parameterized by the cluster module number of G .

Cluster module number

Proof

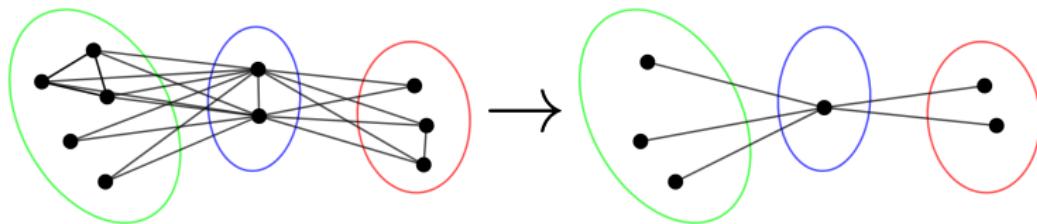
- ① Obtain an optimal cluster module partition of G .



Cluster module number

Proof

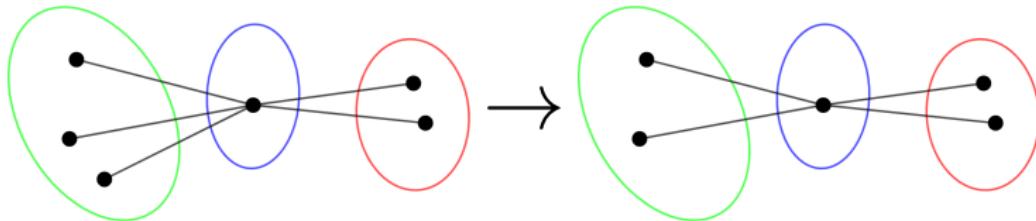
- ① Obtain an optimal cluster module partition of G .
- ② (Lemma 1) Remove all vertices but one of each clique in each part of the cluster module partition.



Cluster module number

Proof

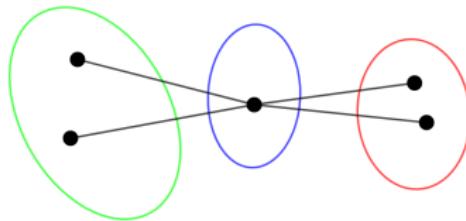
- ① Obtain an optimal cluster module partition of G .
- ② (Lemma 1) Remove all vertices but one of each clique in each part of the cluster module partition.
- ③ (Lemma 2) Remove all vertices until only two of each part remain.



Cluster module number

Proof

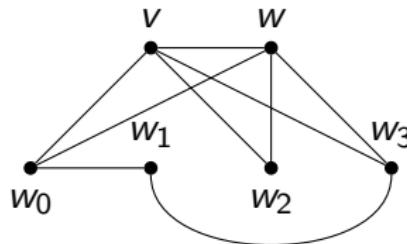
The resulting graph is k -thin if and only if the original graph is k -thin, and it has at most $2 \cdot \text{cm}(G)$ vertices. Thus, this is a **kernel** of size $2 \cdot \text{cm}(G)$.



Neighborhood type

Definition (Neighborhood type)

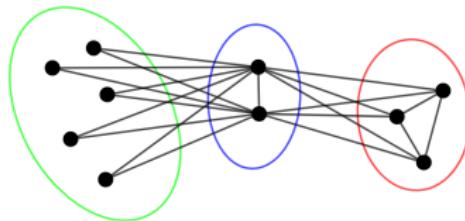
Two vertices v and w have the same *neighborhood type* if $N(v) \setminus \{w\} = N(w) \setminus \{v\}$.



Neighborhood partition

Definition (Neighborhood partition, Neighborhood diversity)

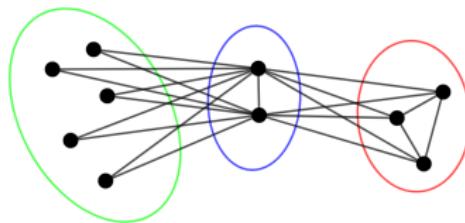
A partition of the vertices of a graph G is a *neighborhood partition* if every two vertices in the same part have the same neighborhood type. The *neighborhood diversity* $\text{nd}(G)$ of G is the minimum number of parts in a neighborhood partition of G .



Neighborhood diversity

Lemma ([4])

Every part of a neighborhood partition is either a clique or an independent set.



Neighborhood diversity

Lemma

$$\text{cm}(G) \leq \text{nd}(G).$$

Neighborhood diversity

Lemma

$$\text{cm}(G) \leq \text{nd}(G).$$

We can then use the same kernelization algorithm as for the cluster module number to obtain a kernel of size at most $2 \cdot \text{nd}(G)$ for THINNESS parameterized by the neighborhood diversity of G .

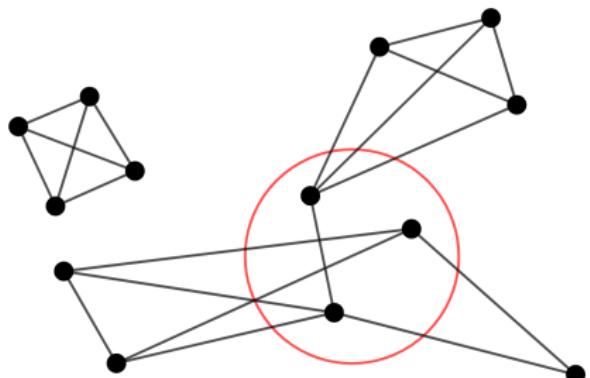
Twin-cover

Definition (Twin-cover)

A subset X of vertices of G is a *twin-cover* if for every edge (u, v) of $E(G)$, either

- one of $\{u, v\}$ belongs to X , or
- u and v are twins, meaning, $N[u] = N[v]$.

The *twin-cover number* $tc(G)$ of G is the minimum size of a twin-cover of G .

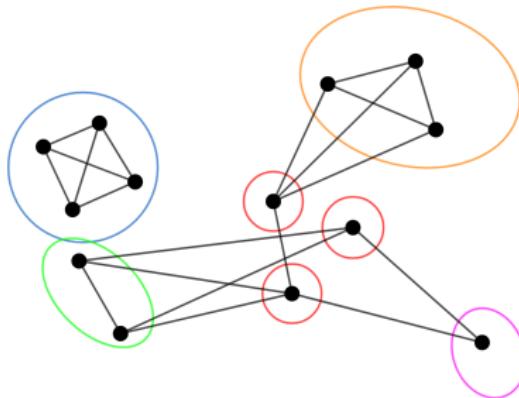


Twin-cover

Lemma

$$\text{cm}(G) \leq 2^{\text{tc}(G)} + \text{tc}(G).$$

Proof: We create $\text{tc}(G)$ parts, one for each vertex in a twin-cover, and then partition the remaining vertices by their **neighborhoods in X**.



Twin-cover

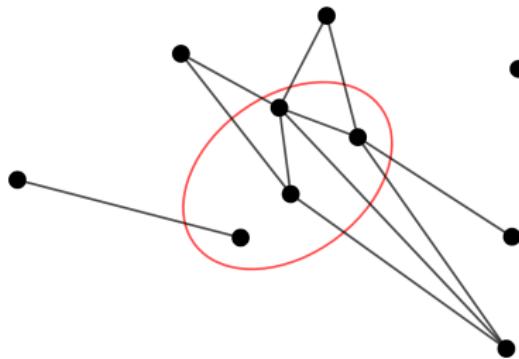
We can again use the same kernelization algorithm to obtain a kernel of size at most $2 \cdot (2^{\text{tc}(G)} + \text{tc}(G))$ for THINNESS parameterized by the twin-cover number of G .

Vertex cover

Definition

A *vertex cover* of a graph G is a subset S of vertices such that every edge of G is incident to at least one vertex in S .

The *vertex cover number* $\text{vc}(G)$ of G is the minimum size of a vertex cover of G .



Vertex cover

Every vertex cover is a twin-cover. We can use the same kernelization algorithm to obtain a kernel of size at most $2 \cdot (2^{\text{vc}(G)} + \text{vc}(G))$ for THINNESS parameterized by the vertex cover number of G .

Kernel size

The kernels when parameterized by cluster module number and neighborhood diversity have **linear size** in the parameter, while the kernels when parameterized by twin-cover and vertex cover number have **exponential size** in the parameter.

Size lower bound

Theorem

Let p be a graph parameter such that

- for every graph G , we have $p(G) \leq |V(G)|$, and
- if H is the disjoint union of two graphs G_1 and G_2 , we have $p(H) \leq \max\{p(G_1), p(G_2)\}$.

Then THINNESS parameterized by p has no polynomial kernel assuming $\text{NP} \not\subseteq \text{coNP/poly}$.

Size lower bound

This does not apply when parameterizing THINNESS by the twin-cover or the vertex cover, but it does apply when parameterizing by

- treewidth,
- bandwidth,
- **thinness**,
- ... and many more.

References

-  **Flavia Bonomo-Braberman, Eric Brandwein, Carolina Lucía Gonzalez, and Agustín Sansone.**
On the thinness of trees.
In Ivana Ljubić, Francisco Barahona, Santanu S. Dey, and A. Ridha Mahjoub, editors, 2022, volume 13526, pages 189–200. Springer, 2022.
-  **Flavia Bonomo-Braberman, Carolina Lucía Gonzalez, Fabiano de S. Oliveira, Moysés S. Sampaio Jr., and Jayme Luiz Szwarcfiter.**
Thinness of product graphs.
312:52–71, 2022.
-  **Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh.**
Parameterized Algorithms.
Springer, 1st edition, 2015.
-  **Michael Lampis.**
Algorithmic meta-theorems for restrictions of treewidth.
64:549–560, 2010.
-  **Yaroslav Shitov.**
Graph thinness, a lower bound and complexity, 2021.
viXra:2112.0157.

Thank you!