Enumeration kernels for Vertex Cover and Feedback Vertex Set

Marin Bougeret
LIRMM, Université de Montpellier, CNRS, Montpellier, France

Guilherme C. M. Gomes
LIRMM, Université de Montpellier, CNRS, Montpellier, France
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, Montpellier, France

Vinicius F. dos Santos
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Enumeration

Enumeration problem

List the set $\text{Sol}(x)$ of all solutions associated with the instance x that satisfy your problem's constraints.

An example - Vertex Cover

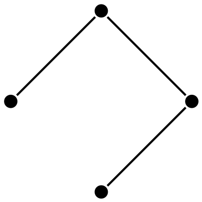
VERTEX COVER
Input: A graph G and an integer k.
Question: Does G have a vertex cover of size at most k?

An example - Vertex Cover

Vertex Cover

Input: A graph G and an integer k.

Question: Does G have a vertex cover of size at most k?

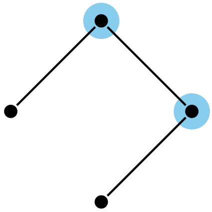


An example - Vertex Cover

Vertex Cover

Input: A graph G and an integer k.

Question: Does G have a vertex cover of size at most k?

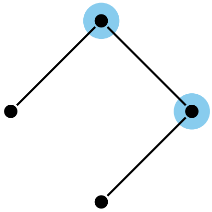


An example - Vertex Cover

Enum Vertex Cover

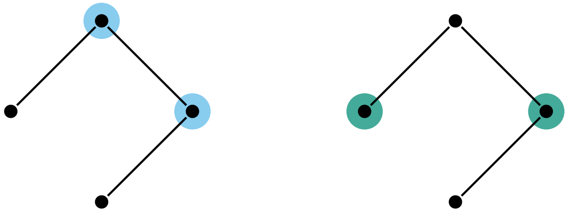
Input: A graph G and an integer k.

Enumerate: All vertex covers of G of size at most k.



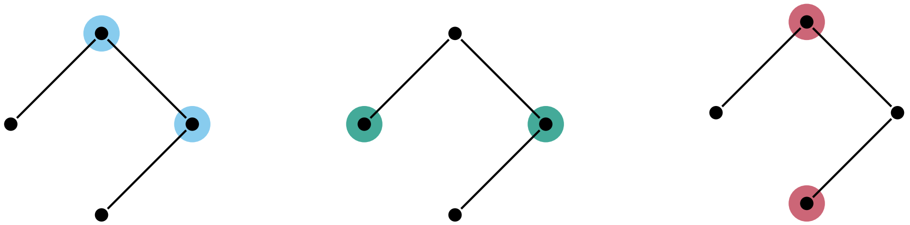
An example - Vertex Cover

Enum Vertex Cover
Input: A graph G and an integer k.
Enumerate: All vertex covers of G of size at most k.



An example - Vertex Cover

Enum Vertex Cover
Input: A graph G and an integer k.
Enumerate: All vertex covers of G of size at most k.



More on enumeration

Applications

Bioinformatics, robotics, optimization, ...

More on enumeration

Applications

Bioinformatics, robotics, optimization, ...

Input-sensitive paradigm

Complexity should be measured by input size only.

More on enumeration

Applications

Bioinformatics, robotics, optimization, ...

Input-sensitive paradigm

Complexity should be measured by input size only. Can't do any meaningful analysis if we have exponentially many solutions.

More on enumeration

Applications

Bioinformatics, robotics, optimization, ...

Input-sensitive paradigm

Complexity should be measured by input size only. Can't do any meaningful analysis if we have exponentially many solutions.

Output-sensitive paradigm

Complexity should be measured by input size and number of solutions.

More on enumeration

Applications

Bioinformatics, robotics, optimization, ...

Input-sensitive paradigm

Complexity should be measured by input size only. Can't do any meaningful analysis if we have exponentially many solutions.

Output-sensitive paradigm

Complexity should be measured by input size and number of solutions.

- Incremental polynomial time: i-th solution of x should be output in $\text{poly}(|x| + i)$.

More on enumeration

Applications

Bioinformatics, robotics, optimization, ...

Input-sensitive paradigm

Complexity should be measured by input size only. Can't do any meaningful analysis if we have exponentially many solutions.

Output-sensitive paradigm

Complexity should be measured by input size and number of solutions.

- Incremental polynomial time: i-th solution of x should be output in $\text{poly}(|x| + i)$.
- Polynomial-delay: time between consecutive outputs in $\text{poly}(|x|)$.

Parameterized complexity for decision

Decision problems & FPT

Each instance x of problem is given with a parameter k, and Π is said to be fixed-parameter tractable if it can be solved in $f(k) \cdot |x|^{\mathcal{O}(1)}$-time.

Preprocessing as kernelization

A kernelization algorithm takes (x, k) as input, runs in polynomial time, and outputs an equivalent instance (y, ℓ) with $|y|, \ell \leq g(k)$.

Theorem

A parameterized problem admits an FPT algorithm $\Leftrightarrow$ it admits a kernel. It is in P $\Leftrightarrow$ $g(k) \in \mathcal{O}(1)$.

Goal of kernelization: minimize $g(k)$.

Parameterized enumeration

FPT-delay

If Π is a parameterized enumeration problem, then FPT-delay is commonly accepted as the "right" notion of tractability:

We want to spend at most $f(k) \cdot |x|^{\mathcal{O}(1)}$-time between consecutive outputs of an instance of Π.

Parameterized enumeration

FPT-delay

If Π is a parameterized enumeration problem, then FPT-delay is commonly accepted as the "right" notion of tractability:

We want to spend at most $f(k) \cdot |x|^{\mathcal{O}(1)}$-time between consecutive outputs of an instance of Π.

Kernelization

??

Enum-kernels

Introduced by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum-kernels

Introduced by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum-kernel

Given (x, k) of Π, kernelization happens in two phases:

Enum-kernels

Introduced by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum-kernel

Given (x, k) of Π, kernelization happens in two phases:

- Compression: Output an equivalent (y, ℓ) of Π in polynomial time with $|y|, \ell \leq g(k)$.

Enum-kernels

Introduced by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum-kernel

Given (x, k) of Π, kernelization happens in two phases:

- Compression: Output an equivalent (y, ℓ) of Π in polynomial time with $|y|, \ell \leq g(k)$.
- Lifting: Given a solution Y of y, output a possibly empty $S_Y \subseteq \mathrm{Sol}(x)$ with $(f(k) \cdot |x|^{\mathcal{O}(1)})$-delay.

Enum-kernels

Introduced by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum-kernel

Given (x, k) of Π, kernelization happens in two phases:

- Compression: Output an equivalent (y, ℓ) of Π in polynomial time with $|y|, \ell \leq g(k)$.
- Lifting: Given a solution Y of y, output a possibly empty $S_Y \subseteq \mathrm{Sol}(x)$ with $(f(k) \cdot |x|^{\mathcal{O}(1)})$-delay. The non-empty S_Y's must form a partition of $\mathrm{Sol}(x)$.

Enum-kernels

Introduced by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum-kernel

Given (x, k) of Π, kernelization happens in two phases:

- Compression: Output an equivalent (y, ℓ) of Π in polynomial time with $|y|, \ell \leq g(k)$.
- Lifting: Given a solution Y of y, output a possibly empty $S_Y \subseteq \mathrm{Sol}(x)$ with $(f(k) \cdot |x|^{\mathcal{O}(1)})$-delay. The non-empty S_Y's must form a partition of $\mathrm{Sol}(x)$.

Theorem (Creignou, Meier, Müller, Schmidt, Vollmer. 2017)

Π *admits an* FPT*-delay algorithm* $\Leftrightarrow$ *it admits an enum-kernel.*

A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.

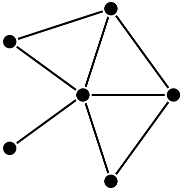
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



$k = 4$

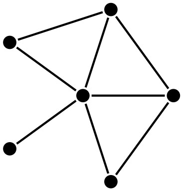
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



$k = 4$

Rule 1

If $v \in V(G)$ has degree $\geq k+1$, remove v and $k \leftarrow k-1$.

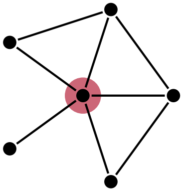
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



$k = 4$

Rule 1

If $v \in V(G)$ has degree $\geq k + 1$, remove v and $k \leftarrow k - 1$.

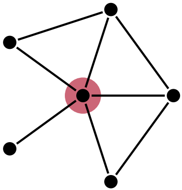
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

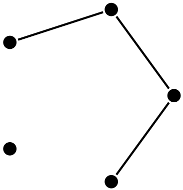
Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



$k = 4$



$k = 3$

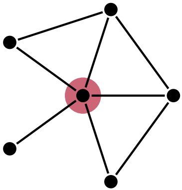
Rule 1

If $v \in V(G)$ has degree $\geq k + 1$, remove v and $k \leftarrow k - 1$.

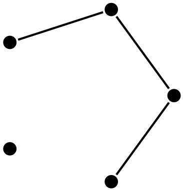
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover
Input: A graph G and an integer k (the parameter).
Enumerate: All vertex covers of G of size at most k.



$k = 4$



$k = 3$

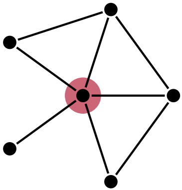
Rule 2

If $v \in V(G)$ has degree 0, remove v.

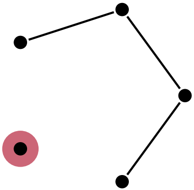
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover
Input: A graph G and an integer k (the parameter).
Enumerate: All vertex covers of G of size at most k.



$k = 4$



$k = 3$

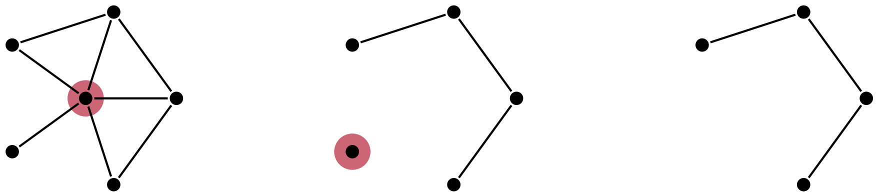
Rule 2

If $v \in V(G)$ has degree 0, remove v.

A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover
Input: A graph G and an integer k (the parameter).
Enumerate: All vertex covers of G of size at most k.



$k = 4$ $k = 3$ $k = 3$

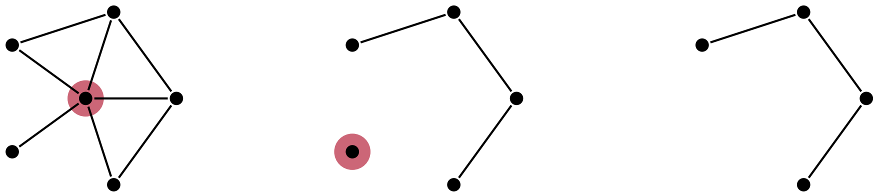
Rule 2

If $v \in V(G)$ has degree 0, remove v.

A k^2 enum-kernel for ENUM VERTEX COVER

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

ENUM VERTEX COVER
Input: A graph G and an integer k (the parameter).
Enumerate: All vertex covers of G of size at most k.



$k = 4$ $k = 3$ $k = 3$

A bounding criterion

No applicable rule $\rightarrow$ max degree k. $|E(G)| > k^2 \rightarrow$ NO-instance.

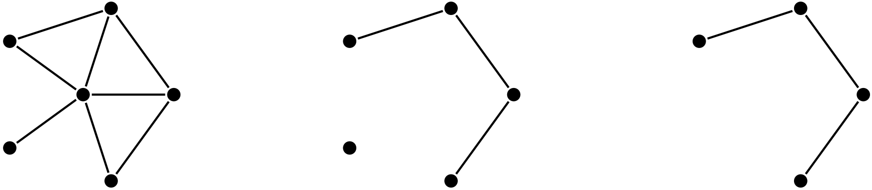
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



$k = 4$ $k = 3$ $k = 3$

Lifting

Take $Y \in \mathrm{Sol}(G', k')$; we never remove vertices from it.

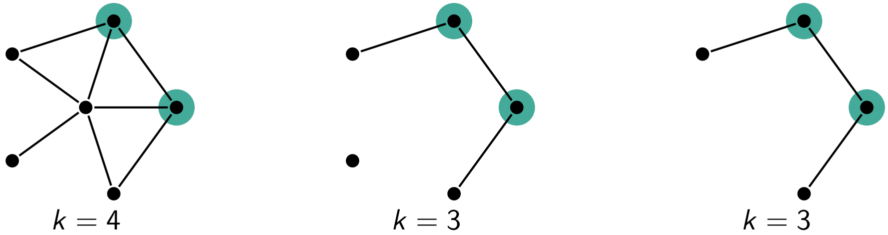
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



Lifting from Rule 2

May add the deleted vertices if $k - |Y| > 0$.

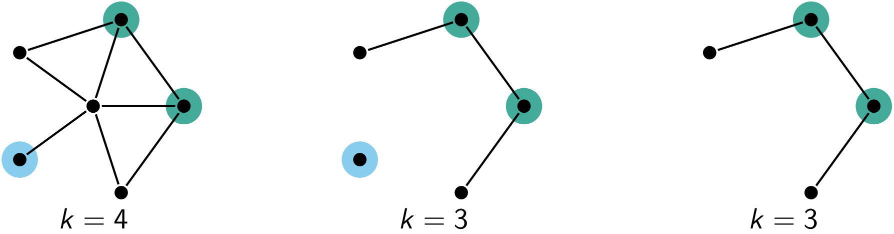
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



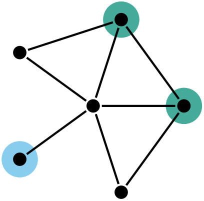
Lifting from Rule 2

May add the deleted vertices if $k - |Y| > 0$.

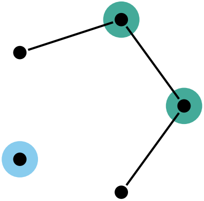
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

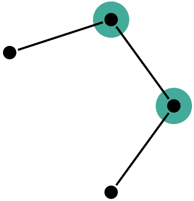
Enum Vertex Cover
Input: A graph G and an integer k (the parameter).
Enumerate: All vertex covers of G of size at most k.



$k = 4$



$k = 3$



$k = 3$

Lifting from Rule 1

Must add the deleted vertices; can do so since $k > |Y|$.

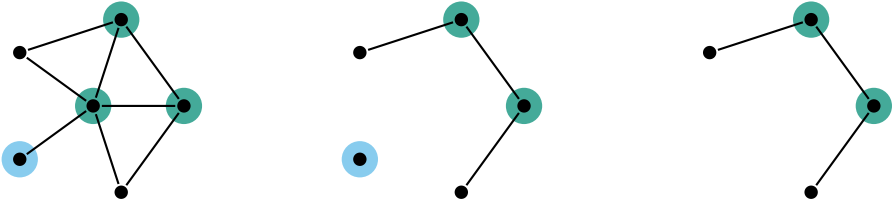
A k^2 enum-kernel for Enum Vertex Cover

Observed by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Enum Vertex Cover

Input: A graph G and an integer k (the parameter).

Enumerate: All vertex covers of G of size at most k.



$k = 4$ $k = 3$ $k = 3$

Lifting from Rule 1

Must add the deleted vertices; can do so since $k > |Y|$.

Another look at enum-kernels

Theorem (Creignou, Meier, Müller, Schmidt, Vollmer. 2017)

Π *admits an* FPT*-delay algorithm* $\Leftrightarrow$ *it admits an enum-kernel.*

Another look at enum-kernels

Theorem (Creignou, Meier, Müller, Schmidt, Vollmer. 2017)

Π *admits an* FPT-*delay algorithm* $\Leftrightarrow$ *it admits an enum-kernel.*

Question

Is polynomial-delay equivalent to a constant-size enum-kernel?

Another look at enum-kernels

Theorem (Creignou, Meier, Müller, Schmidt, Vollmer. 2017)

Π *admits an* FPT-*delay algorithm* $\Leftrightarrow$ *it admits an enum-kernel.*

Question

Is polynomial-delay equivalent to a constant-size enum-kernel?

Not really, as pointed out by [Golovach, Komusiewicz, Kratsch, Le. 2022]

FPT-delay is equivalent to a constant-size enum-kernel.

Another look at enum-kernels

Theorem (Creignou, Meier, Müller, Schmidt, Vollmer. 2017)

Π *admits an FPT-delay algorithm* ⇔ *it admits an enum-kernel.*

Question

Is polynomial-delay equivalent to a constant-size enum-kernel?

Not really, as pointed out by [Golovach, Komusiewicz, Kratsch, Le. 2022]

FPT-delay is equivalent to a constant-size enum-kernel.

A new model for enumeration kernels needed to be introduced.

Polynomial-delay (PD) kernels

New model introduced by [Golovach, Komusiewicz, Kratsch, Le. 2022]

Polynomial-delay (PD) kernel

Given (x, k), kernelization happens in two phases:

Polynomial-delay (PD) kernels

New model introduced by [Golovach, Komusiewicz, Kratsch, Le. 2022]

Polynomial-delay (PD) kernel

Given (x, k), kernelization happens in two phases:

- Compression: Output an equivalent instance (y, ℓ) in polynomial-time with $|y|, \ell \leq g(k)$.

Polynomial-delay (PD) kernels

New model introduced by [Golovach, Komusiewicz, Kratsch, Le. 2022]

Polynomial-delay (PD) kernel

Given (x, k), kernelization happens in two phases:

- Compression: Output an equivalent instance (y, ℓ) in polynomial-time with $|y|, \ell \leq g(k)$.
- Lifting: Given $Y \in \mathrm{Sol}(y)$, output a non-empty $S_Y \subseteq \mathrm{Sol}(x)$ with $\mathrm{poly}(|x| + |y| + k + \ell)$-delay.
  The S_Y's must partition $\mathrm{Sol}(x)$.

Polynomial-delay (PD) kernels

New model introduced by [Golovach, Komusiewicz, Kratsch, Le. 2022]

Polynomial-delay (PD) kernel

Given (x, k), kernelization happens in two phases:

- Compression: Output an equivalent instance (y, ℓ) in polynomial-time with $|y|, \ell \leq g(k)$.
- Lifting: Given $Y \in \mathrm{Sol}(y)$, output a non-empty $S_Y \subseteq \mathrm{Sol}(x)$ with $\mathrm{poly}(|x| + |y| + k + \ell)$-delay.
  The S_Y's must partition $\mathrm{Sol}(x)$.

Theorem (Golovach, Komusiewicz, Kratsch, Le. 2022)

Problem Π admits a PD kernel $\Leftrightarrow$ it admits an FPT-delay algorithm.
Moreover, $g(k) \in \mathcal{O}(1) \Leftrightarrow \Pi$ is solvable with polynomial-delay.

Known polynomial PD kernels

Problem	Parameter	Kernel size

Known polynomial PD kernels

Problem	Parameter	Kernel size
Enum Vertex Cover	Vertex cover	k^2

Kernel found by [Creignou, Meier, Müller, Schmidt, Vollmer. 2017]

Known polynomial PD kernels

Problem	Parameter	Kernel size
Enum Vertex Cover	Vertex cover	k^2
Enum Matching Cut	Vertex cover	k^2
	Neigh. diversity	k
	Feedback edge set	k

Kernels found by [Golovach, Komusiewicz, Kratsch, Le. 2022]

Known polynomial PD kernels

Problem	Parameter	Kernel size
Enum Vertex Cover	Vertex cover	k^2
Enum Matching Cut	Vertex cover	k^2
	Neigh. diversity	k
	Feedback edge set	k
Enum d-Cut	Vertex cover	k^2
	Neigh. diversity	k
	Clique partition	k^{d+2}

Kernels found by [Komusiewicz, Majumdar. 2023]

Known polynomial PD kernels

Problem	Parameter	Kernel size
Enum Vertex Cover	Vertex cover	k^2
Enum Matching Cut	Vertex cover	k^2
	Neigh. diversity	k
	Feedback edge set	k
Enum d-Cut	Vertex cover	k^2
	Neigh. diversity	k
	Clique partition	k^{d+2}
Enum Matching Multicut	Vertex cover	k^2
	Dist. co-cluster	k^2

Kernels found by [Gomes, Juliano, Martins, dos Santos. IPEC 2024]

Known polynomial PD kernels

Problem	Parameter	Kernel size
Enum Vertex Cover	Vertex cover	k^2
Enum Matching Cut	Vertex cover	k^2
	Neigh. diversity	k
	Feedback edge set	k
Enum d-Cut	Vertex cover	k^2
	Neigh. diversity	k
	Clique partition	k^{d+2}
Enum Matching Multicut	Vertex cover	k^2
	Dist. co-cluster	k^2
Enum Long Path	Vertex cover	k^2
	Dissociation	k^3
	Dist. to clique	k^3

Kernels found by [Komusiewicz, Majumdar, Sommer. 2025]

Known polynomial PD kernels

Problem	Parameter	Kernel size
Enum Vertex Cover	Vertex cover	k^2
Enum Matching Cut	Vertex cover	k^2
	Neigh. diversity	k
	Feedback edge set	k
Enum d-Cut	Vertex cover	k^2
	Neigh. diversity	k
	Clique partition	k^{d+2}
Enum Matching Multicut	Vertex cover	k^2
	Dist. co-cluster	k^2
Enum Long Path	Vertex cover	k^2
	Dissociation	k^3
	Dist. to clique	k^3

These were all the known PD kernels.

Our results

Theorem

ENUM VERTEX COVER *admits a PD kernel with at most* $2k$ *vertices when parameterized by the solution size.*

Our results

Theorem

ENUM VERTEX COVER *admits a PD kernel with at most* $2k$ *vertices when parameterized by the solution size.*

Surprising fact: no enumeration kernels were known for ENUM FEEDBACK VERTEX SET.

Our results

Theorem

ENUM VERTEX COVER *admits a PD kernel with at most* $2k$ *vertices when parameterized by the solution size.*

Surprising fact: no enumeration kernels were known for ENUM FEEDBACK VERTEX SET.

Theorem

ENUM FEEDBACK VERTEX SET *admits a PD kernel with* $\mathcal{O}(k^3)$ *vertices when parameterized by the solution size.*

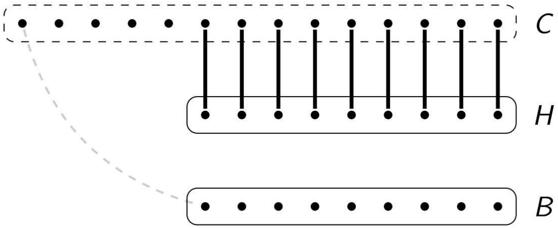
Sketch of the linear kernel for Enum Vertex Cover

Sketch of the linear kernel for Enum Vertex Cover

Crown decomposition of a graph G

A partition (C, H, B) of $V(G)$ such that:

- C is an independent set.
- H separates C and B.
- there is an H-saturating matching between H and C;



Sketch of the linear kernel for Enum Vertex Cover

Crown decomposition of a graph G

A partition (C, H, B) of $V(G)$ such that:

- C is an independent set.
- H separates C and B.
- there is an H-saturating matching between H and C;

Theorem (Nemhauser and Trotter, 1975 + Chlebík and Chlebíková, 2008)

Let G be a graph without isolated vertices and at least $2k + 1$ vertices. Then, there is a polynomial-time algorithm that either:

- *Decides that no vertex cover of size at most k exists.*
- *Or finds a crown decomposition of G.*

The compression step of the kernel is very easy

The compression step of the kernel is very easy

Rule 1

Let (G, k) be the input instance. If v is an isolated vertex of G:

- Remove v from G.
- Keep k unchanged.

The compression step of the kernel is very easy

Rule 1

Let (G, k) be the input instance. If v is an isolated vertex of G:

- Remove v from G.
- Keep k unchanged.

Rule 2

If G has a crown decomposition (C, H, B):

- Remove C, H from G.
- Set $k \leftarrow k - |H|$.

The compression step of the kernel is very easy

Rule 1

Let (G, k) be the input instance. If v is an isolated vertex of G:

- Remove v from G.
- Keep k unchanged.

Rule 2

If G has a crown decomposition (C, H, B):

- Remove C, H from G.
- Set $k \leftarrow k - |H|$.

Rule 3

If $k < 0$, conclude that we are dealing with no-instance.

The compression step of the kernel is very easy

Rule 1

Let (G, k) be the input instance. If v is an isolated vertex of G:

- Remove v from G.
- Keep k unchanged.

Rule 2

If G has a crown decomposition (C, H, B):

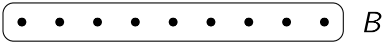
- Remove C, H from G.
- Set $k \leftarrow k - |H|$.

Rule 3

If $k < 0$, conclude that we are dealing with no-instance.

The resulting graph has at most $2k$ vertices.

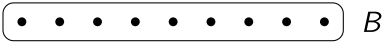
The lifting algorithm is the hard part!



The lifting algorithm is the hard part!

Lifting

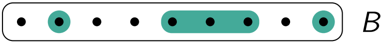
Let $Y \in \mathrm{Sol}(G[B], k - |H|)$.



The lifting algorithm is the hard part!

Lifting

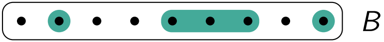
Let $Y \in \mathrm{Sol}(G[B], k - |H|)$.



The lifting algorithm is the hard part!

Lifting

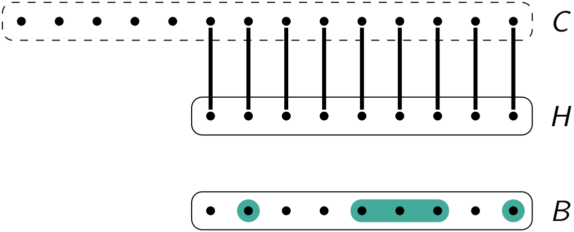
Let $Y \in \text{Sol}(G[B], k - |H|)$. We only add vertices of $H \cup C$.



The lifting algorithm is the hard part!

Lifting

Let $Y \in \mathrm{Sol}(G[B], k - |H|)$. We only add vertices of $H \cup C$.

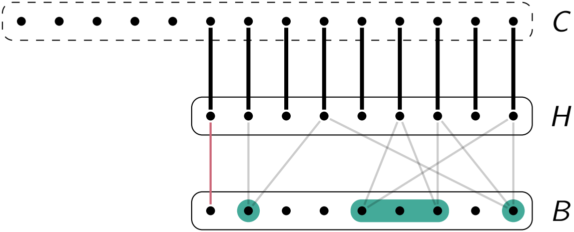


Add to Y any $v \in H \cap N(B)$ incident to an edge uncovered by Y.

The lifting algorithm is the hard part!

Lifting

Let $Y \in \mathrm{Sol}(G[B], k - |H|)$. We only add vertices of $H \cup C$.

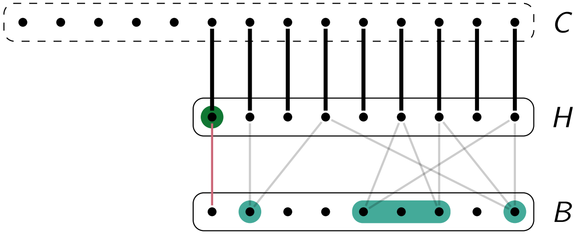


Add to Y any $v \in H \cap N(B)$ incident to an edge uncovered by Y.

The lifting algorithm is the hard part!

Lifting

Let $Y \in \mathrm{Sol}(G[B], k - |H|)$. We only add vertices of $H \cup C$.



Add to Y any $v \in H \cap N(B)$ incident to an edge uncovered by Y.

A linear kernel for Enum Vertex Cover

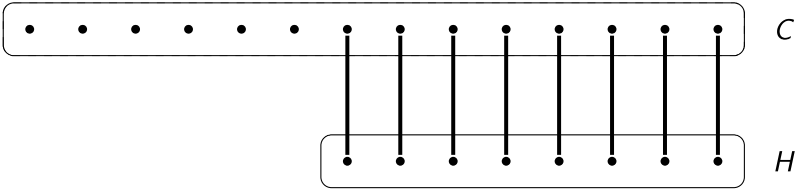
Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.

A linear kernel for Enum Vertex Cover

Task

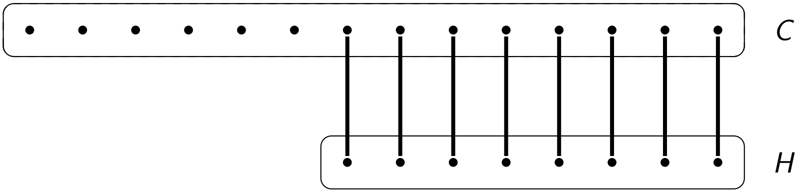
Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



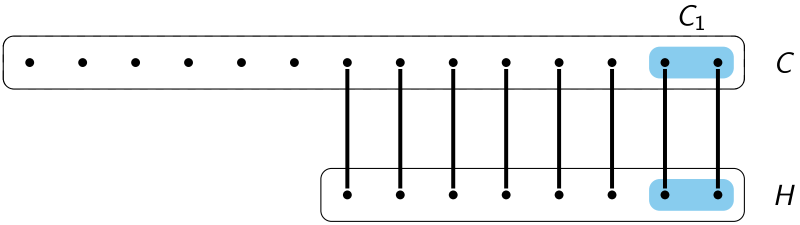
Step 1

Choose E_1 to be the only matching edges with both endpoints in Y.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



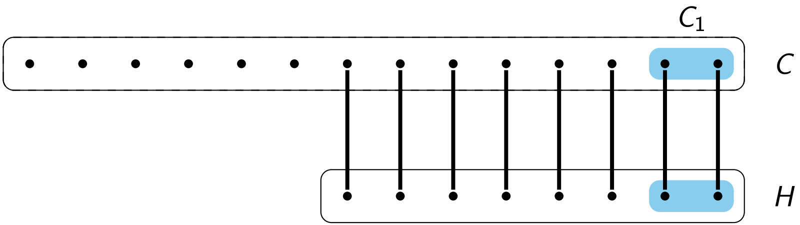
Step 1

Choose E_1 to be the only matching edges with both endpoints in Y.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



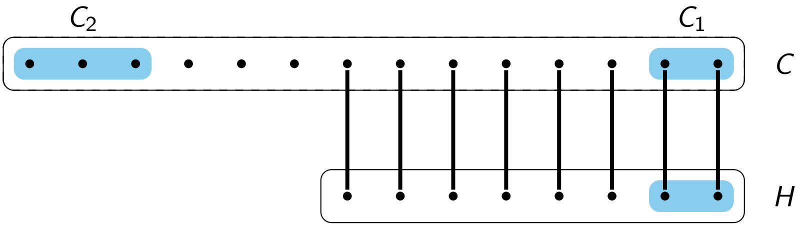
Step 2

Choose $d_2 \leq s - |E_1|$ unmatched vertices and add them to Y.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



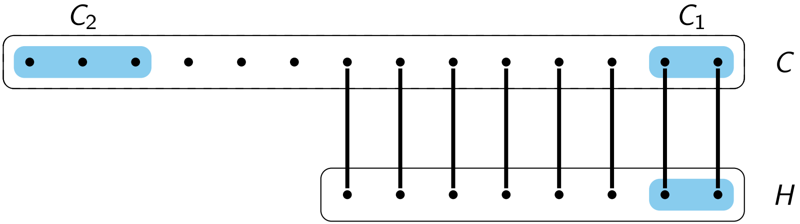
Step 2

Choose $d_2 \leq s - |E_1|$ unmatched vertices and add them to Y.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



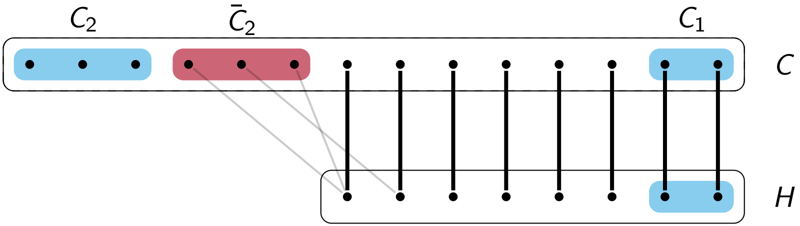
Step 3

Unmatched vertices $\notin C_2$ force vertices of H to be picked.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



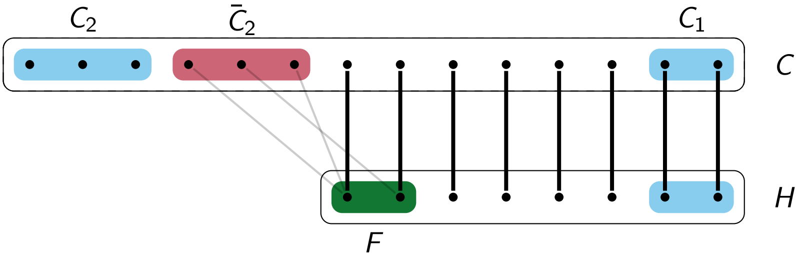
Step 3

Unmatched vertices $\notin C_2$ force vertices of H to be picked.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



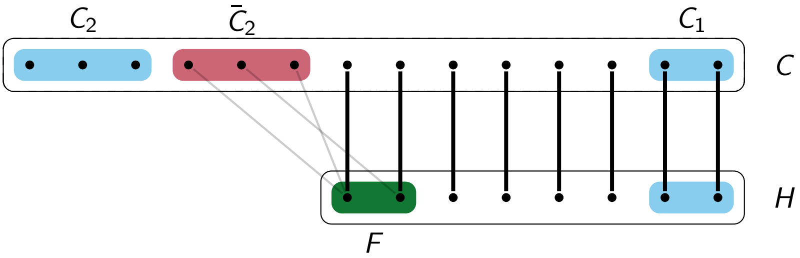
Step 3

Unmatched vertices $\notin C_2$ force vertices of H to be picked.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



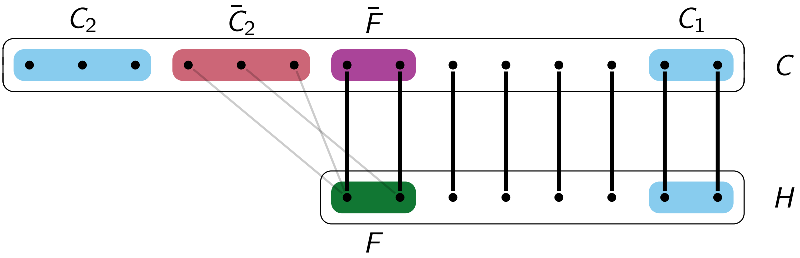
Step 4

Which force its matched vertices to be excluded.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



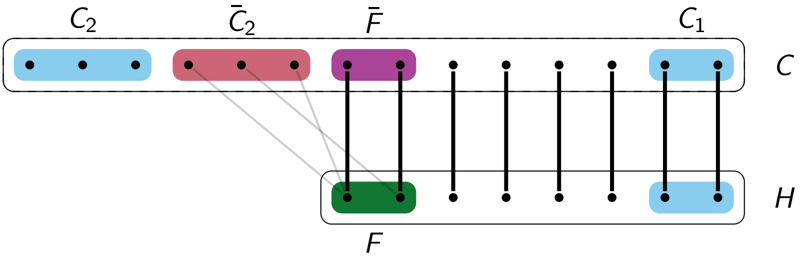
Step 4

Which force its matched vertices to be excluded.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



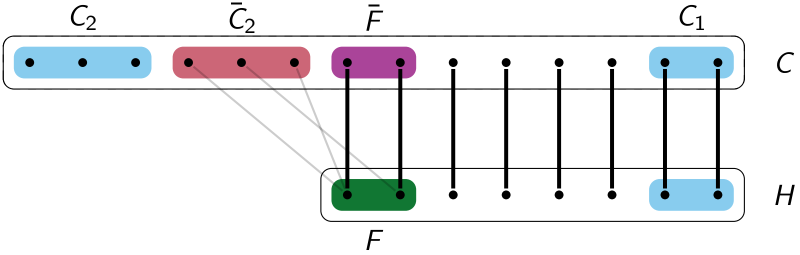
Step 4

Which force its matched vertices to be excluded.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



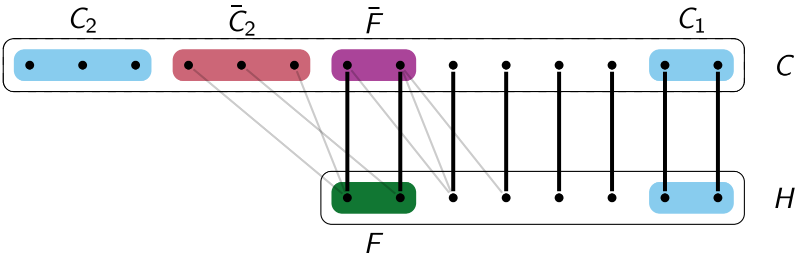
Step 5

Which force its other neighbors to be picked.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



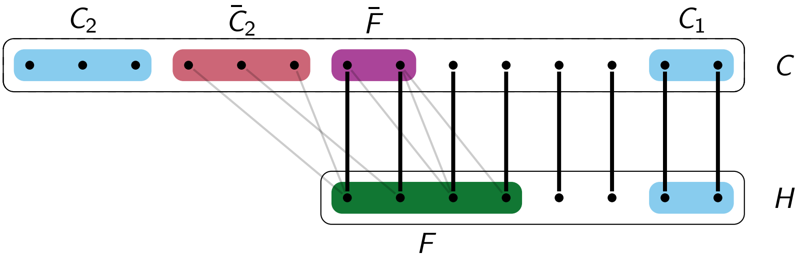
Step 5

Which force its other neighbors to be picked.

A linear kernel for ENUM VERTEX COVER

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



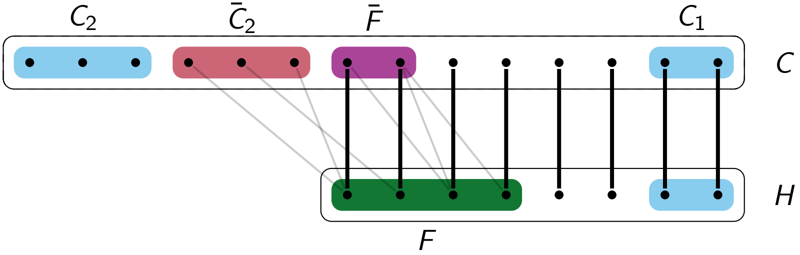
Step 5

Which force its other neighbors to be picked.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



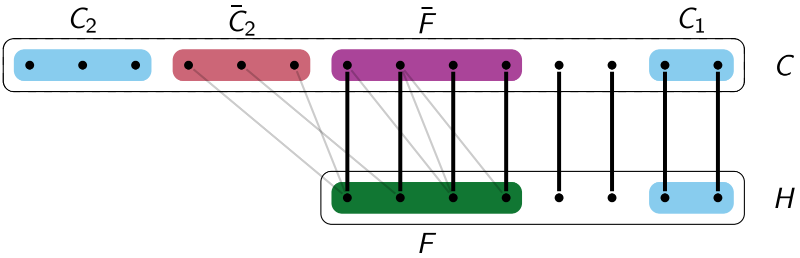
Step 6

Keep going until we don't have to grow F anymore.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



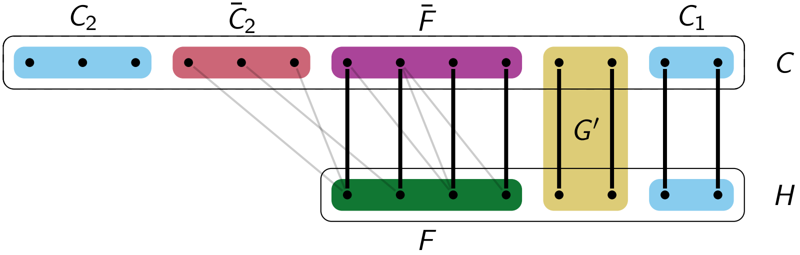
Step 6

Keep going until we don't have to grow F anymore.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



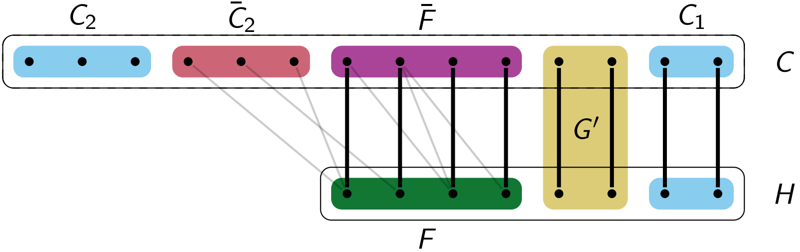
Step 6

Keep going until we don't have to grow F anymore.

A linear kernel for Enum Vertex Cover

Task

Enumerate all vertex covers of $G[H \cup C \setminus Y]$ of size $\leq k - |Y| = |H| + s$.



What remains

Enumerate vertex covers of G' of size $|V(G')|/2$.

A linear kernel for Enum Vertex Cover

After a long branching algorithm on G' and even longer correctness argument...

A linear kernel for Enum Vertex Cover

After a long branching algorithm on G' and even longer correctness argument...

Theorem

Enum Vertex Cover *admits a PD kernel with at most* $2k$ *vertices when parameterized by the solution size.*

A linear kernel for ENUM VERTEX COVER

After a long branching algorithm on G' and even longer correctness argument...

Theorem

ENUM VERTEX COVER *admits a PD kernel with at most* $2k$ *vertices when parameterized by the solution size.*

Observation

Requiring that every solution of the compressed instance outputs some solution of (G, k) significantly complicates the lifting procedure.

Sketch of the cubic kernel for Enum Feed. Vert. Set

Theorem

Enum Feedback Vertex Set *admits a PD kernel with* $\mathcal{O}(k^3)$ *vertices when parameterized by the solution size.*

Sketch of the cubic kernel for Enum Feed. Vert. Set

Theorem

Enum Feedback Vertex Set *admits a PD kernel with* $\mathcal{O}(k^3)$ *vertices when parameterized by the solution size.*

- Inspired by quadratic kernel for decision version. [Thomassé. 2010]

Sketch of the cubic kernel for Enum Feed. Vert. Set

Theorem

Enum Feedback Vertex Set *admits a PD kernel with* $\mathcal{O}(k^3)$ *vertices when parameterized by the solution size.*

- Inspired by quadratic kernel for decision version. [Thomassé. 2010]
- Namely, design rules to lower and upper bound the degree of the graph, and then observe that $|V(G)| = \mathcal{O}(\Delta(G) \cdot \mathrm{fvs}(G))$.

Sketch of the cubic kernel for Enum Feed. Vert. Set

Theorem

Enum Feedback Vertex Set admits a PD kernel with $\mathcal{O}(k^3)$ vertices when parameterized by the solution size.

- Inspired by quadratic kernel for decision version. [Thomassé. 2010]
- Namely, design rules to lower and upper bound the degree of the graph, and then observe that $|V(G)| = \mathcal{O}(\Delta(G) \cdot \mathrm{fvs}(G))$.
- But the core ingredient (finding a 2-expansion in an auxiliary graph) seems to fail for the enumeration version.

Sketch of the cubic kernel for Enum Feed. Vert. Set

Theorem

Enum Feedback Vertex Set admits a PD kernel with $\mathcal{O}(k^3)$ vertices when parameterized by the solution size.

- Inspired by quadratic kernel for decision version. [Thomassé. 2010]
- Namely, design rules to lower and upper bound the degree of the graph, and then observe that $|V(G)| = \mathcal{O}(\Delta(G) \cdot \mathrm{fvs}(G))$.
- But the core ingredient (finding a 2-expansion in an auxiliary graph) seems to fail for the enumeration version.
- **Main reason**: cannot deal with double edges s.t. there exists some solution using one of their endpoints (but maybe not all of them).

Sketch of the cubic kernel for ENUM FEED. VERT. SET

Theorem

ENUM FEEDBACK VERTEX SET *admits a PD kernel with* $\mathcal{O}(k^3)$ *vertices when parameterized by the solution size.*

- Inspired by quadratic kernel for decision version. [Thomassé. 2010]
- Namely, design rules to lower and upper bound the degree of the graph, and then observe that $|V(G)| = \mathcal{O}(\Delta(G) \cdot \mathrm{fvs}(G))$.
- But the core ingredient (finding a 2-expansion in an auxiliary graph) seems to fail for the enumeration version.
- **Main reason**: cannot deal with double edges s.t. there exists some solution using one of their endpoints (but maybe not all of them).
- As a result, we obtain a maximum degree of $\mathcal{O}(k^2)$ (instead of $\mathcal{O}(k)$).

Further research

Further research

- Parameterized enumeration lower bounds:

Further research

- Parameterized enumeration lower bounds:
  - How to rule out the existence of an FPT-delay algorithm when the decision version is in FPT?

Further research

- Parameterized enumeration lower bounds:
  - How to rule out the existence of an FPT-delay algorithm when the decision version is in FPT?
  - How to rule out the existence of polynomial enumeration kernels when the decision version has a polynomial kernel?

Further research

- Parameterized enumeration lower bounds:
  - How to rule out the existence of an FPT-delay algorithm when the decision version is in FPT?
  - How to rule out the existence of polynomial enumeration kernels when the decision version has a polynomial kernel?
  - Fine grained (S)ETH-like lower bounds?

Further research

- Parameterized enumeration lower bounds:
    - How to rule out the existence of an FPT-delay algorithm when the decision version is in FPT?
    - How to rule out the existence of polynomial enumeration kernels when the decision version has a polynomial kernel?
    - Fine grained (S)ETH-like lower bounds?
- Is there a $\mathcal{O}\left(k^2\right)$ PD kernel for Enum Feedback Vertex Set?

Further research

- Parameterized enumeration lower bounds:
  - How to rule out the existence of an FPT-delay algorithm when the decision version is in FPT?
  - How to rule out the existence of polynomial enumeration kernels when the decision version has a polynomial kernel?
  - Fine grained (S)ETH-like lower bounds?
- Is there a $\mathcal{O}(k^2)$ PD kernel for Enum Feedback Vertex Set?
- More examples of enumeration kernels for classical problems.

Further research

- Parameterized enumeration lower bounds:
  - How to rule out the existence of an FPT-delay algorithm when the decision version is in FPT?
  - How to rule out the existence of polynomial enumeration kernels when the decision version has a polynomial kernel?
  - Fine grained (S)ETH-like lower bounds?
- Is there a $\mathcal{O}(k^2)$ PD kernel for Enum Feedback Vertex Set?
- More examples of enumeration kernels for classical problems.

Recall: observation

Requiring that every solution of the compressed instance outputs some solution of (G, k) significantly complicates the lifting procedure.