A More Versatile Model for Enumerative Kernelization: A Case Study for Vertex Cover

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

Abstract

Enumerative kernelization is a relatively recent and promising area sitting at the intersection of parameterized complexity and enumeration algorithms, with two main models being proposed. The first, known as enum-kernels and due to Creignou et al. [Theory Comput. Syst., 2017], was too permissive, leading to constant-sized kernels for every problem solvable with FPT-delay. To remedy this, Golovach et al. [J. Comput. Syst. Sci., 2022] proposed the polynomial-delay enumeration kernelization model that, while addressing the shortcoming of the previous one, appears to be too strict, which we believe is a central reason for the slow development that the area has enjoyed so far. In this paper, we propose a new model for enumeration kernels, which we have called polynomial-delay (PD) kernels. It is more flexible than Golovach et al.'s kernels while still preserving their qualities; informally, it allows us to ignore "bad" solutions of the compressed instance when producing the solution set of the input instance, but still requires that the "good" solutions are lifted with polynomial-delay. After discussing the main properties of our model, we design a generic framework for vertex-subset problems to adapt decision kernels into PD kernels of the same size. We showcase our model's increased versatility and the expressive power of our framework on the Enum Vertex Cover problem, where we want to list all vertex covers of size at most k of a given graph. In particular, we manage to generalize the kernelization dichotomy by Bougeret et al. [SIAM J. Discrete Math., 2022] about the existence of polynomial kernels for Vertex Cover parameterized by the vertex deletion distance to a minor-closed graph class, as well as the solution size and feedback vertex number parameterizations. The second one, in particular, is significantly simpler than the kernel designed by Bougeret et al. [IPEC, 2025], requiring only a few lines for its lifting algorithm. Beyond our framework, we also show how to generalize to the enumeration setting the kernel of Bougeret et al. [Algorithmica, 2019] for the vertex-deletion distance to c-treedepth.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Kernelization, Enumeration, Vertex Cover, Structural parameterization

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.76

Related Version *Full Version*: https://arxiv.org/abs/2604.23419

Funding *Guilherme C. M. Gomes*: Funded by the European Union, project PACKENUM, grant number 101109317. Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.
Ignasi Sau: French project ELIT (ANR-20-CE48-0008-01).

1 Introduction

Enumeration problems are central objects in both practical and theoretical computer science, with the main example of the former being the branch-and-bound method [36] (and its relatives), universally used in integer programming solvers. Instead of answering a question in either the positive or the negative, in this class of problems the goal is to generate all witnesses, or *solutions*, that satisfy the given problem's constraints, or decide that no solution exists; we denote by $\mathsf{Sol}(x)$ the solution set of an instance x of some fixed problem of interest. Typically, the number of solutions will be exponential in the input size n, and thus *input-sensitive* algorithms, whose running times are only analyzed with respect to n, are not powerful enough tools to capture all nuances intrinsic to enumeration problems; in particular, *input-polynomial* algorithms, whose running times are of the form $n^{\mathcal{O}(1)}$, are inadequate models of efficient enumeration. Consequently, enumeration algorithms are usually analyzed in the *output-sensitive* context, i.e., their running times depend on both n and the size of the output; in this setting, the most popular classes of efficiently solvable enumeration problems are those that admit *incremental-polynomial time* [13, 33] or *polynomial-delay* [3, 23, 32] algorithms, named IncP and DelayP, respectively, for which the relation $\mathsf{DelayP} \subsetneq \mathsf{IncP}$ holds [12, 41]. In IncP, the time taken to output the i-th solution should be polynomial in $i + n$; whereas in DelayP, we want that, between startup and the first output (the precalculation), any two outputs, and between the last output and termination (postcalculation), the running time is bounded by $\mathsf{poly}(n)$. These classes were extended by Creignou et al. [16], which established links between enumeration complexity and the polynomial hierarchy of Stockmayer [40] to derive a hierarchy of hard enumeration problems and appropriate notions of reductions between them. Incremental-polynomial time and polynomial-delay algorithms are unfeasible goals in many interesting cases; in particular, enumeration problems for which the existence of a single solution is an NP-hard problem do not admit such algorithms unless $\mathsf{P} = \mathsf{NP}$.

Orthogonally to enumeration, the theory of parameterized complexity gained significant traction as a means of coping with NP-hardness, with thousands of papers and some books [18, 22, 26, 27, 39] dedicated to the subject. In this framework, algorithms are analyzed according to both the total input size $n = |x|$ and a *parameter* of interest $k = \kappa(x)$, that is also part of the input, with the efficient algorithms, known as *fixed-parameter tractable* (FPT), being those of running time $f(k) \cdot n^{\mathcal{O}(1)}$, for some computable function f. This increase in computational power allows for the existence of efficient algorithms for several important NP-hard problems, such as Vertex Cover and Feedback Vertex Set, when parameterized by the solution size. Complementarily, a rich lower bound theory capable of showing fine grained limits to this broader notion of tractability has also been developed, offering further insights between parameterized and non-parameterized decision problems. An equivalent definition of tractability in this framework, and central to our work, is that of *kernelization*. A *kernel* is a polynomial-time algorithm that, given an instance (x, k), outputs an equivalent instance (y, ℓ) such that $y, \ell \leq g(k)$ for some computable function g, known as the *size* of the kernel. Kernels can be seen as mathematically guaranteed *pre-processing* algorithms, and as a way to explain the tremendous success that simple clean-up routines have in real-world applications, such as in integer optimization. Polynomial size kernels are of particular interest, as they typically result in significant reductions in input size.

A very successful case study in kernelization is that of Vertex Cover, arguably the most fundamental problem in parameterized complexity. Abu-Khzam et al. [2] presented the first kernel with $\mathcal{O}(k)$ vertices for the problem when parameterized by the solution size k, using the then-novel concept of *crown decompositions* to improve on the previously best

known kernel with $\mathcal{O}(k^2)$ vertices, implied by a result of Buss and Goldsmith [10]. Vertex Cover has been the paradigmatic problem in the very active area of kernelization with *structural parameters*, that is, parameters that take into account structural properties of the input, and that can be much smaller than the solution size. The ultimate goal is to unveil the "smallest" parameters for which the problem admits a polynomial kernel. This research program was triggered by the breakthrough result of Jansen and Bodlaender [31] showing the existence of a kernel with $\mathcal{O}(t^3)$ vertices for Vertex Cover parameterized by the size t of a given feedback vertex set. Not long after, it became a textbook example to show that no polynomial kernel exists when parameterizing by the treewidth of the input graph [5, 18], and so the main question was now how deep into the graph parameter hierarchy one could go, particularly when the parameterization was on the vertex-deletion distance to minor-closed graph classes. In 2019, Bougeret and Sau [9] took another step in this direction by presenting a polynomial kernel under the parameterization of vertex-deletion distance to a graph of treedepth c, for some constant c; see also [28, 30, 37]. Finally, Bougeret et al. [8] proved that, under standard complexity hypotheses, Vertex Cover admits a polynomial kernel when parameterized by the vertex-deletion distance to some minor-closed graph class $\mathcal{F}$ if and only if $\mathcal{F}$ has bounded *bridgedepth*, a new parameter they introduced to this end.

Given the growth of the parameterized complexity field, it is no surprise then that there have been mounting efforts to extend it to the enumeration setting [4, 11, 17, 20, 21, 25, 29, 34, 35, 38], with *input*-FPT, *incremental*-FPT, and FPT-*delay* having their expected definitions; we also highlight the work of Creignou et al. [17] as being particularly influential in this endeavor. Satisfactory notions of kernelization for enumeration problems, which we broadly call *enumerative kernelization*, were not as quickly defined, even though kernelization itself did play a significant role in several works. In the early 2000s, Fernau [25] presented an input-FPT algorithm for the enumeration of *minimum* vertex covers when parameterized by the solution size k, one of the first parameterized enumeration algorithms in the literature. Interestingly for our work, Fernau's algorithm uses a kernelization strategy based on the classical Buss kernel [10, 22] to argue that we can reduce this problem to the enumeration of minimum vertex covers in a graph with $\mathcal{O}(k^2)$ vertices; the algorithm, however, uses standard parameterized decision complexity language and mechanisms, and so does not offer many other insights beyond its own existence. A few years later, Damaschke [20] investigated subset minimization problems, i.e., where the goal is to enumerate every subset of size at most k of a given ground set that satisfies the desired property. For example, Enum Vertex Cover is the problem where, given a graph G and integer k, the goal is to enumerate every subset of vertices of size at most k that hits all edges of G. For this type of problem, Damaschke introduced the notion of *full kernels*: in a subset minimization problem, a full kernel is a set that contains the union of all minimal solutions of size at most k. To the best of our knowledge, this is the first model of enumerative kernelization, even if it is not a general purpose one. Nevertheless, Damaschke applied this new formalism to Enum c-Hitting Set parameterized by the maximum solution size k, thus showing an input-FPT algorithm under this parameterization. In a subsequent paper, Damaschke [21] successfully applied full kernels to Enum Cluster Edge-Editing: the problem of enumerating all ways of making at most k editions, with each edition being an edge addition or removal, such that every connected component of the resulting graph is a clique, when parameterized by k. All of these results, in the end, imply that the target problems admit input-FPT algorithms; in this direction, only very recently did Golovach et al. [29] define *fully-polynomial enumeration kernels* (FPE kernels), which are equivalent to the existence of an input-FPT algorithm, much like FPT algorithms and kernels are equivalent in the decision world; we will not discuss input-sensitive enumeration algorithms or kernels further, as our focus is on delay-based algorithms.

Output-sensitive parameterized enumeration was little explored until Creignou et al. [17] defined *enum-kernels*: a general purpose kernelization framework for parameterized enumeration problems that is applicable to a problem if and only if it admits an FPT-delay algorithm. Intuitively, the kernel has two parts: the first, $\mathcal{A}_1$, is the *compression algorithm* and has the same constraints as a decision kernel, while the second $\mathcal{A}_2$, known as the *lifting algorithm*, receives the input (x, k), the output (y, ℓ) of $\mathcal{A}_1$, and a solution $Y \in \mathsf{Sol}(y)$, and must output, with $(f(k+\ell) \cdot \mathsf{poly}(|x| + |y| + |Y|))$-delay, where f is a computable function, a (possibly empty) subset $\mathcal{S}_Y \subseteq \mathsf{Sol}(x)$; additionally, the $\mathcal{S}_Y$'s must be disjoint and their union must be precisely $\mathsf{Sol}(x)$. The authors then showcased their proposed kernelization model on Enum Vertex Cover parameterized by the maximum solution size (called All-Vertex-Cover in [17]) by proving that the classical Buss kernel [10, 22] is an enum-kernel. This model, however, has a fatal limitation. As shown by Golovach et al. [29], a parameterized enumeration problem admits an FPT-delay algorithm if and only if it admits an enum-kernel of *constant* size. As such, there is little point in developing small enum-kernels besides as a certificate that the target problems has an FPT-delay algorithm. To overcome this issue, Golovach et al. proposed a new model for enumerative kernelization, which they called *polynomial-delay enumeration kernel*: instead of FPT-delay, the lifting algorithm is restricted to *polynomial-delay* but must always output at least one solution. With this alteration, they are still capable of proving that their kernels and FPT-delay algorithms are equivalent, while gaining the critical property that a problem admits a kernel of constant size if and only if it admits a polynomial-delay algorithm.

In the same paper that introduces the kernelization model, Golovach et al. [29] provide these types of kernels for several enumeration variants of the Matching Cut problem under different structural parameterizations; it is also indirectly discussed how the Buss kernel is, in fact, a polynomial-delay enumeration kernel for Enum Vertex Cover of quadratic size. Their work has since been followed by some further examples of these kernels. In particular, Komusiewicz and Majumdar [34] studied structural parameterizations of Enum d-Cut, whose decision version is a direct generalization of Matching Cut. Simultaneously, Gomes et al. [11] generalized Matching Cut in a new direction, known as Matching Multicut, as well as some results of Golovach et al. [29] to Enum Matching Multicut. Outside of Matching Cut-adjacent problems, we are only aware of two works on the subject. The first, by Komusiewicz et al. [35], is on an enumeration variant of Longest Path under structural parameterizations. In the second one, which only came out very recently, Bougeret et al. [7] developed polynomial-sized polynomial-delay enumeration kernels for Enum Vertex Cover and Enum Feedback Vertex Set under their natural parameters k; for the first, they developed a lengthy lifting algorithm for the classical crown decomposition-based kernel to obtain a kernel with $2k$ vertices, while for the latter they designed a novel $\mathcal{O}(k^3)$ kernel.

While the enum-kernels of Creignou et al. [17] had too much lifting power, Golovach et al.'s [29] polynomial-delay enumeration kernels have proven quite complex to be designed, with even what should be the simplest example, Enum Vertex Cover, demanding long and cumbersome lifting algorithms. By closely examining the latter model's examples, much of the difficulty seems to originate in the requirement that *every* solution Y is lifted to a *non-empty* set S_Y. Informally, this imposes the constraint that no "noise" is introduced in $\mathsf{Sol}(y)$ when compressing x (typically, it implies that $|\mathsf{Sol}(y)| \leq |\mathsf{Sol}(x)|$). This is a critical limitation in our view, as it implies that polynomial-delay enumeration kernels cannot handle failure states, a common strategy in enumeration algorithms. For example, in the ordered generation framework of Eiter et al. [24], initially developed for the Minimal Hypergraph Transversal Enumeration problem, known as Trans-Enum in the

enumeration community, bad intermediate solutions exist and are pruned iteratively, and these may only be found deep within the algorithm's recursion tree. Furthermore, the e-, D-, and I-reductions used by Creignou et al. [16] to characterize their hierarchy of hard enumeration problems heavily rely on the possibility that the reduced instance has solutions unsuited to be lifted back to solutions of the input.

Our contributions. We propose a new enumerative kernelization model that sits between Creignou et al.'s and Golovach et al.'s; while it requires that lifting be performed with polynomial-delay (as in the latter), it also handles failure states, by allowing that some solutions of the compressed instance be ignored by the lifting procedure (as in the former). As evidenced by our results, this allowance leads to simpler algorithms while having the same nice theoretical guarantees of polynomial-delay enumeration kernels. Our new enumerative kernelization model, which we call *polynomial-delay kernel*, PD kernel for short, can also be seen as a generalization of Golovach et al.'s polynomial-delay enumeration kernels; for convenience we rename the latter as *strong PD kernels*. We begin by showing some fundamental properties of our model. In particular, we prove that its existence indeed corresponds to FPT-delay algorithms and that the size of the kernel is a meaningful metric to optimize (cf. Theorem 7). To showcase our model's versatility, we introduce a generic framework for graph vertex-subsets problems, which allows for the systematic translation of decision kernels to their enumeration counterparts. This framework is based on Conditions 1–3, which encapsulate sufficient properties that, if satisfied by a decision kernel and the problem Π at hand, immediately imply a PD kernel of the same size for the problem of enumerating all solutions of Π; this is a highly desired property, as it is natural to try to leverage the extensive body of knowledge of decision kernelization to the enumeration setting. Intuitively, Condition 1 asks that the solution set of the input can be partitioned into similarity classes and that the kernelization preserves enough information to distinguish these classes; Condition 2 asks for an efficient algorithm that decides if an output solution should be a *canonical solution* for a given class of input solutions; finally, Condition 3 asks for a polynomial-delay algorithm that takes a canonical solution and lists all input solutions of the associated class. We prove the sufficiency of these conditions in Theorem 9. As an evidence of our model's power, we apply the proposed framework in our main technical contribution: proving that the dichotomy for minor-closed graph classes of decision kernelization [8] also holds in the enumeration setting. That is, we show that, under standard complexity hypotheses, Enum Vertex Cover admits a polynomial-sized PD kernel when parameterized by the size of a given modulator X of G to a minor-closed graph class $\mathcal{F}$ (i.e., $G \setminus X$ has no minor forbidden by $\mathcal{F}$) if and only if $\mathcal{F}$ has bounded bridgedepth; this is formally stated in Theorem 1.

▶ **Theorem 1.** *Let $\mathcal{F}$ be a minor-closed graph class and assume* NP $\not\subseteq$ coNP/poly. Enum Vertex Cover *parameterized by the size of a given modulator to $\mathcal{F}$ admits a PD kernel of polynomial size if and only if $\mathcal{F}$ has bounded bridgedepth.*

The positive part of our proof of Theorem 1 essentially boils down to showing that, modulo a very simple rule applied at the beginning of the compression step to identify trivial instances, Conditions 1–3 are readily satisfied by the kernel developed by Bougeret et al. [8] to prove the decision version of Theorem 1. The hardness part of the proof follows immediately from the matching decision kernelization lower bound for Vertex Cover proved in [8].

Beyond Theorem 1, to illustrate how our model and framework can be used, we engage in the more interesting journey of reproving some of the most influential kernelization results for Vertex Cover that led to the analogous dichotomy theorem. Namely, we also revisit

the parameterizations by: the size of the maximum size of the wanted vertex cover [2, 10], the feedback vertex number [31], and the vertex-deletion distance to graphs of treedepth at most c [9]. With the exception of the latter, all of our proofs are based on the developed framework. This is due to the fact that the decision kernel of Bougeret and Sau [9] relies on a bikernelization strategy; while it is possible to generalize our framework to encompass this type of strategy, it becomes overly cumbersome, and it is much simpler to prove that it can be accomplished in an ad-hoc manner. We do so by introducing what we believe to be the first bikernelization algorithm for an enumeration problem, as well as appropriate notions of reductions between enumeration problems that enable such an algorithm to exist.

Further research. We hope that the added versatility of the introduced model will further stimulate research on enumerative kernelization. We are interested in investigating enumeration versions of classical problems, such as Feedback Vertex Set, and determining if and how other decision kernels can be adapted to our model. Specifically for Feedback Vertex Set, we are interested in the existence of quadratic PD kernels, which would directly improve the cubic strong PD kernel recently given by Bougeret et al. [7]. We also point out that a lower bound theory for enumerative kernelization (and for enumerative parameterized complexity as a whole!) would be of extreme benefit to the area. Currently, we rely on lower bounds for related parameterized *decision* problems to obtain enumeration lower bounds for both FPT-delay and PD kernelization algorithms. It is highly unlikely, however, that this strategy is powerful enough to distinguish which parameterized enumeration problems are tractable or not, and which admit kernels of polynomial size or not.

Organization. In Section 2 we introduce our new model for enumerative kernelization and prove that it indeed satisfies the natural desirable properties that one may expect. In Section 3 we present the generic framework that allows to systematically lift decision kernels to enumeration kernels, based on Conditions 1–3. To illustrate the versatility of our model, in particular via the application of the generic framework, in Section 4 we obtain a simple linear kernel for Enum Vertex Cover parameterized by the solution size (*much* simpler than the linear strong PD kernel recently obtained by Bougeret et al. [7]). Due to space constraints, and to fully present the kernel of Section 4, the more technical proofs of Sections 2 and 3, detailed preliminaries, and all of our kernels for structural parameterizations are deferred to the full version. We primarily follow the graph-theoretic notation of [6] and the parameterized complexity notation of [18].

2 A new model for enumerative kernelization

Let U be a finite set and define 2^U as the *powerset* of U, i.e., the set of all subsets of U. A collection $\mathcal{P} = \{P_1, \ldots, P_k\}$ is a *(k-)partition* of U if $\mathcal{P} \subseteq 2^U \setminus \{\emptyset\}$, $\bigcup_{i \in [k]} P_i = U$, and for every $P_i, P_j \in \mathcal{P}$, we have $P_i \cap P_j \neq \emptyset$ if and only if $i = j$. We start with basic definitions.

▶ **Definition 2** (Parameterized enumeration problem (Creignou et al. [17], Golovach et al. [29])). *A* parameterized enumeration problem *over a finite alphabet* Σ *is a tuple* $\Pi = (L, \mathsf{Sol}, \kappa)$, *shorthanded by* Π^κ, *such that:*

i. $L \subseteq \Sigma^\star$ *is a decidable language.*
ii. $\mathsf{Sol} \colon \Sigma^\star \mapsto 2^{\Sigma^\star}$ *is a computable function such that, for every* instance $x \in \Sigma^\star$, $\mathsf{Sol}(x)$ *is finite and we have* $\mathsf{Sol}(x) \neq \emptyset$ *if and only if* $x \in L$.
iii. $\kappa \colon \Sigma^\star \mapsto \mathbb{N}$, *is the* parameterization.

An instance x *of* Π^κ *is a* no-instance *if* $\mathsf{Sol}(x) = \emptyset$, *and is a* yes-instance *otherwise.*

▶ **Definition 3** (FPT-delay algorithm (Creignou et al. [17], Golovach et al. [29]))**.** *Let Π^κ be a parameterized enumeration problem. A* fixed-parameter-tractable-delay algorithm*, or simply* FPT-delay *algorithm, is an algorithm $\mathcal{A}$ that, given an instance x of Π^κ and $k = \kappa(x)$, outputs $\mathsf{Sol}(x)$ such that, the precalculation time, the time between any two consecutive outputs, and the postcalculation time, are bounded by $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f. If $f(k) \in \mathcal{O}(1)$, then we say that $\mathcal{A}$ is a* polynomial-delay *algorithm.*

We are now ready to formally define our kernelization model.

▶ **Definition 4** (Polynomial-delay kernel)**.** *Let Π^κ be a parameterized enumeration problem. A* polynomial-delay kernel*,* PD kernel *for short, is a pair of algorithms $\mathcal{A}_1, \mathcal{A}_2$ that, given an instance x of Π^κ and $\kappa(x)$:*

- *Algorithm $\mathcal{A}_1$ runs in $\mathsf{poly}(|x| + \kappa(x))$-time and outputs an instance y of Π^κ satisfying $\kappa(y), |y| \leq f(\kappa(x))$, with f a computable function, and $\mathsf{Sol}(y) \neq \emptyset$ if and only if $\mathsf{Sol}(x) \neq \emptyset$;*
- *Algorithm $\mathcal{A}_2$ receives x, $y = \mathcal{A}_1(x, \kappa(x))$, $\kappa(x)$, $\kappa(y)$, some $Y \in \mathsf{Sol}(y)$, and outputs, with polynomial-delay g on its total input size, $S_Y \subseteq \mathsf{Sol}(x)$. Moreover, we require that $\{S_Y \mid Y \in \mathsf{Sol}(y), S_Y \neq \emptyset\}$ is a partition of $\mathsf{Sol}(x)$.*

We call $\mathcal{A}_1$ the compression algorithm *and $\mathcal{A}_2$ the* lifting algorithm*. We say that f is the* size *of the kernel, and g is its* delay*. The pair $(\mathcal{A}_1, \mathcal{A}_2)$ is a* strong PD kernel *if, for every $Y \in \mathsf{Sol}(y)$, $\mathcal{A}_2$ never outputs the empty set.*

▶ Remark 5. Strong PD kernels and polynomial-delay enumeration kernels, as defined by Golovach et al. [29], are the same objects. Moreover, in their definition, the $\mathsf{Sol}(x) \neq \emptyset \Leftrightarrow \mathsf{Sol}(y) \neq \emptyset$ condition is implied by $\mathcal{A}_2$ only being allowed to output non-empty sets.

Observe that, *a priori*, we could alter the definition of the compression algorithm to only require that $\mathsf{Sol}(x) \neq \emptyset$ implies $\mathsf{Sol}(y) \neq \emptyset$; the lifting algorithm could still be used to correctly identify that $\mathsf{Sol}(x) = \emptyset$ even if $\mathsf{Sol}(y) \neq \emptyset$. This, however, would configure a situation that to us seems unnatural, as we highlight in Remark 6.

▶ Remark 6. Take the problem of enumerating all independent sets of size at least t of a graph G when parameterized by size of a modulator X to graphs of treewidth two , with (G, X, t) as its input. Outputting $(G[X], X, 0)$ is enough: given $A \in \mathsf{Sol}(G[X], X, 0)$ it suffices to use Courcelle's Theorem [15] and flashlight search [42] to check if $G \setminus (N(A) \cup X)$ has an independent set of appropriate size and list all of the appropriate ones. However, it is believed that no kernel exists for the decision version of this problem [19].

Our next theorem shows that PD kernels and FPT-delay algorithms are equivalent. Our proof is similar to the one provided by Golovach et al. [29] for strong PD kernels, but must account for the case where the output instance has more solutions than the input.

▶ **Theorem 7** ($\star$)**.** *A parameterized enumeration problem $\Pi^\kappa_{\mathsf{enum}}$ admits an* FPT*-delay algorithm if and only if it admits a PD kernel. Moreover, $\Pi^\kappa_{\mathsf{enum}}$ can be solved with polynomial-delay if and only if it admits a constant-sized PD kernel.*

In [35], it was proved that every strong PD kernel admits FPT precalculation time and polynomial-delay; intuitively, their result guarantees that searching for small strong PD kernels with fast lifting algorithms is a worthwhile pursuit. We generalize their result with Remark 8. This coincides with the proof of Creignou et al. [17] that every problem solvable by an FPT-delay algorithm can be solved with FPT precalculation time and polynomial-delay.

▶ Remark 8 ($\star$). Let $\Pi^\kappa_{\mathsf{enum}} = (L, \mathsf{Sol}, \kappa)$ be a parameterized enumeration problem such that there exists a $T(|x|)$-time algorithm $\mathcal{F}$ to compute $\mathsf{Sol}(x)$, where x is an instance of $\Pi^\kappa_{\mathsf{enum}}$. If $\Pi^\kappa_{\mathsf{enum}}$ admits a PD kernel of size f and delay g, then $\Pi^\kappa_{\mathsf{enum}}$ admits an algorithm with $T(f(\kappa(x))) \cdot g(|x| + \kappa(x) + 2f(\kappa(x)))$ precalculation time and delay g.

In light of Theorem 7 and Remark 8, we note that, if $\Pi^{\kappa}_{\mathsf{enum}}$ admits a polynomial-delay algorithm $\mathcal{P}$, then $f(k) \in \mathcal{O}(1)$, which implies that $T(f(\kappa(x))) \cdot g(|x| + \kappa(x) + 2f(\kappa(x))) \in \mathcal{O}(g(|x|))$; that is, after we kernelize the input into y, we obtain the following polynomial-delay algorithm: *(i)* run $\mathcal{P}$ until termination to obtain a list $\mathsf{Sol}(y)$ ($\mathcal{O}(1)$); *(ii)* for each $S \in \mathsf{Sol}(y)$, run the lifting algorithm until termination or first output, with L being the list of solutions that fall in the latter case ($\mathcal{O}(g(|x|))$); *(iii)* finally, for each surviving solution in L, run the lifting algorithm itself. Since $|\mathsf{L}| \in \mathcal{O}(1)$, step *(iii)* runs with polynomial-delay g. We also point out that *(ii)* is the reason for the $g(\cdot)$ precalculation overhead in Remark 8.

3 A framework for translating decision kernels into PD kernels

While decision kernelization has a large literature from which to draw when designing a new kernel, the same is yet untrue for enumerative kernelization. As such, it is highly desired to understand when one can translate a kernel for a decision problem to one of its enumeration counterparts. In this section, we take a first step in this direction, identifying three sufficient conditions that allow the design of a PD kernel from an existing decision kernel. We concern ourselves only with *graph vertex-subset problems*, i.e., problems where the solutions are precisely subsets of vertices. While this may seem restrictive, several cornerstone problems in parameterized complexity belong to this class; as examples, we point to both Vertex Cover and Clique. For this section, let Π^{κ} be a parameterized graph vertex-subset problem with parameterization κ and $\Pi^{\kappa}_{\mathsf{enum}}$ be the problem of enumerating all solutions of Π^{κ}.

Formally, both Π^{κ} and $\Pi^{\kappa}_{\mathsf{enum}}$ have as input the triple $(G, \mathcal{I}, k)$, where G is the graph of interest, $k = \kappa(G, \mathcal{I})$ is the parameter, and $\mathcal{I}$ is any other meaningful problem input, e.g. $\mathcal{I}$ could include a feedback vertex set of G and/or the threshold size t of the desired solution (for instance, of an independent set). To derive a PD kernel for $\Pi^{\kappa}_{\mathsf{enum}}$, it is necessary that a kernel exists for Π^{κ}, as a kernel for the former implies one for the latter. In this spirit, Conditions 1–3 are a set of sufficient properties that enable us to derive a kernel for $\Pi^{\kappa}_{\mathsf{enum}}$ from a kernel to Π^{κ}. Condition 1 is the most technical one, but intuitively we ask that the Π^{κ} kernel preserve some key information to the enumeration process, which we capture by the concepts of *core* of the kernel and its *good traces*. Condition 2 allows us to efficiently separate the good traces from the bad ones; moreover, it allows us to identify a representative solution for each good trace. Finally, given a good trace Y of the core in the original instance, Condition 3 allows us to list all solutions that intersect its core precisely at Y. We refer to Figure 1 for a diagram depicting these interactions.

▶ **Condition 1.** *Π^{κ} admits a compression algorithm $\mathcal{A}_1$ for which the following holds.*

1. *Given $(G, \mathcal{I}, k)$, $\mathcal{A}_1$ outputs an equivalent $(H, \mathcal{L}, \ell)$ with $|H| \leq |G|$.*
2. *Let $\Lambda_H \subseteq V(H)$, $\lambda \colon \Lambda_H \to V(G)$ be an injection, and $\Lambda_G = \lambda(\Lambda_H) = \bigcup_{v \in \Lambda_H} \{\lambda(v)\}$, i.e., λ is a bijection from Λ_H to Λ_G. We say that (Λ_H, λ) is a* core *for $\mathcal{A}_1$ if, for every $Y_G \subseteq \Lambda_G$ for which there exists $S \in \mathsf{Sol}(G, \mathcal{I}, k)$ with $S \cap \Lambda_G = Y_G$, there exists $S' \in \mathsf{Sol}(H, \mathcal{L}, \ell)$ with $Y_H = S' \cap \Lambda_H$ and $\lambda(Y_H) = Y_G$. Moreover, Λ_H and λ can be computed in $\mathsf{poly}(|G|, k, |\mathcal{I}|, |H|, \ell, |\mathcal{L}|)$-time.*

   All such $Y_H \subseteq \Lambda_H$ and $\lambda(Y_H) \subseteq \Lambda_G$ that satisfy Item 2 are the good traces *of the core.*

We remark that, if in a kernel for Π^{κ} we have that $V(H) \subseteq V(G)$ (which is the case for all kernels of this paper except for one parameterized of Theorem 1) and take λ as the identity function, Condition 1 just asks for the existence of a set $\Lambda_H \subseteq V(H)$ such that each of its subsets that is contained in a solution of $(G, \mathcal{I}, k)$ must also belong to some solution of $(H, \mathcal{L}, \ell)$.

▶ **Condition 2.** *Let $\mathcal{A}_1$ be the algorithm of Condition 1, $(G, \mathcal{I}, k)$ be an input of $\Pi^{\kappa}_{\mathsf{enum}}$, and $(H, \mathcal{L}, \ell)$ be the output of $\mathcal{A}_1(G, \mathcal{I}, k)$. There exists a* choosing algorithm *$\mathcal{A}_c$ that:*

1. *Runs in $\mathsf{poly}(|G|, k, |H|, \ell)$-time.*
2. *Given $(G, \mathcal{I}, k, H, \mathcal{L}, \ell)$ and a single $S' \in \mathsf{Sol}(H, \mathcal{L}, \ell)$, $\mathcal{A}_c$ returns either* yes *or* no.
3. *For every good trace $Y_H \subseteq \Lambda_H$, there is a unique $S_{Y_H} \in \mathsf{Sol}(H, \mathcal{L}, \ell)$ with $S_{Y_H} \cap \Lambda_H = Y_H$ and $\mathcal{A}_c(G, \mathcal{I}, k, H, \mathcal{L}, \ell, S_{Y_H})$ returning* yes*; such an S_{Y_H} is the* canonical solution *for Y_H.*
4. *For every $Y_H \subseteq \Lambda_H$ which is not a good trace, and every $S' \in \mathsf{Sol}(H, \mathcal{L}, \ell)$ with $S' \cap \Lambda_H = Y_H$, then $\mathcal{A}_c(G, \mathcal{I}, k, H, \mathcal{L}, \ell, S')$ returns* no.

Note that Item 3 of Condition 2 implies that for any good trace Y_H and any $S' \in \mathsf{Sol}(H, \mathcal{L}, \ell) \setminus \{S_{Y_H}\}$, $\mathcal{A}_c(G, \mathcal{I}, k, H, \mathcal{L}, \ell, S')$ returns no.

▶ **Condition 3.** *Let $(G, \mathcal{I}, k)$ be an input of $\Pi^{\kappa}_{\mathsf{enum}}$, $(H, \mathcal{L}, \ell)$ be the output of $\mathcal{A}_1(G, \mathcal{I}, k)$, and Λ_G be defined as in Condition 1. There exists an* enumeration algorithm *$\mathcal{A}_e$ that, given a good trace $Y_G \subseteq \Lambda_G$, $\mathcal{A}_e$ produces the set $\{S \in \mathsf{Sol}(G, \mathcal{I}, k) \mid S \cap \Lambda_G = Y_G\}$ with $\mathsf{poly}(|G|, k, \mathcal{I}, |H|, \ell, \mathcal{L}, |Y_G|)$-delay.*

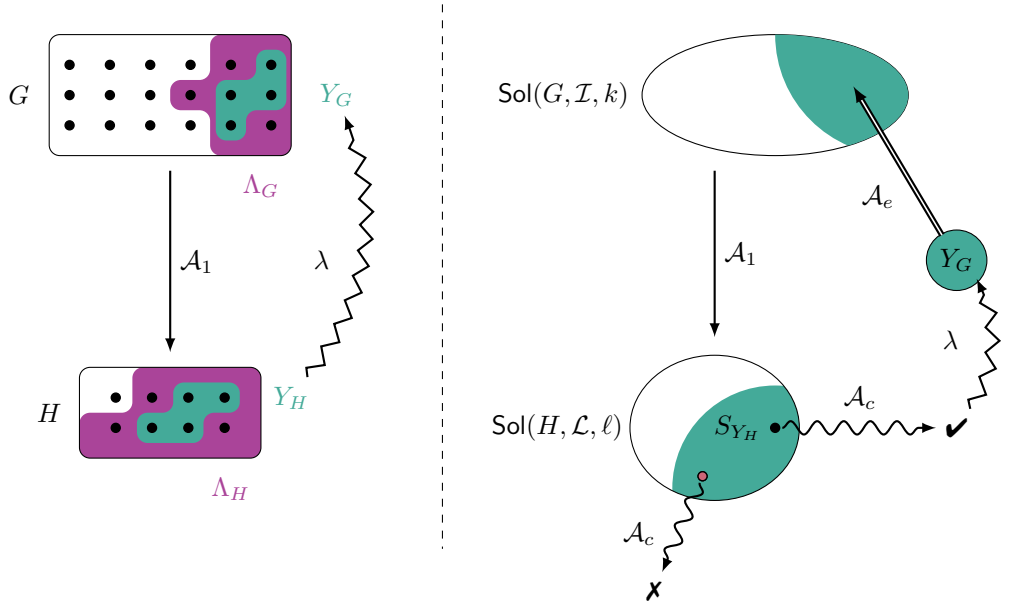

Figure 1 Interactions between the components of Conditions 1–3 on the input and output and on the solutions and solution sets. The lower purple-shaded area indicates the set Λ_H of the output graph H, and the upper area $\Lambda_G = \lambda(\Lambda_H)$; teal-shaded areas correspond to the good traces $Y_H \subseteq \Lambda_H$ and $Y_G = \lambda(Y_H)$ and to the solutions with Y_H or Y_G as their trace. Straight arrows represent the compression algorithm given by Condition 1 on both the input and solution sets; wavy arrows the choosing algorithm of Condition 2; zigzagging arrows the λ function of the kernel's core; and the double arrow the action of the enumeration algorithm of Condition 3 upon the good trace Y_G.

Note that, by combining all three conditions, we have that $\{\{S \in \mathsf{Sol}(G, \mathcal{I}, k) \mid S \cap \Lambda_G = \lambda(Y)\} \mid Y$ is a good trace of $\Lambda_H\}$ is a partition of $\mathsf{Sol}(G, \mathcal{I}, k)$. Our conditions illustrate the two main challenges when working with enumerative kernelization. Namely, having a large core is helpful, as it becomes easy to preserve good traces and lift them back to the input instance; however, we want to compress the instance as much as possible, so, in particular, we must work with a core that is much smaller than the input's.

Before proceeding to the main theorem of this section, we highlight that a similar set of conditions can be obtained for strong PD kernels, but there are two main challenges in specifying such a collection. The first is on the compression algorithm: besides preserving all the good traces, the kernel of Condition 1 would also have to exclude *every* non-good trace, which is a feature rarely found in decision kernels; in particular, when applying reduction rules, we typically restrict ourselves to only preserving some maximal/minimal solutions. The second is on the lifting algorithm: by requiring that it always produces some output, we must also be able to partition the set of solutions with a same trace; this puts additional strain on the algorithm, making its design significantly more complicated.

▶ **Theorem 9** (⋆). *Let Π^κ be parameterized decision graph vertex-subset problem and $\Pi^\kappa_{\mathsf{enum}}$ its parameterized enumeration version. If there exist $\mathcal{A}_1, \mathcal{A}_e, \mathcal{A}_c$ that satisfy Conditions 1–3 and $\mathcal{A}_1$ is a kernel, then $\Pi^\kappa_{\mathsf{enum}}$ admits a PD kernel of the same size as $\mathcal{A}_1$ and delay bounded by the maximum between the delay of $\mathcal{A}_e$ and the running time of $\mathcal{A}_c$.*

As our kernels rely on reduction rules, we prove that they respect Condition 1 in order to leverage Theorem 9. The core of a PD kernel is what links its compression and lifting parts, which allows for a clean decoupling of these phases and simplifies the design of the kernel. A reduction rule is *safe* if the corresponding compression algorithm satisfies the first item of Condition 1, and it is *sound* if it satisfies the second. For most reduction rules we prove only their soundness, overall shortening our proofs. When both safeness and soundness are needed, e.g., when we introduce a novel reduction rule, we prove them separately. Lemma 10 shows that algorithms satisfying Condition 1 compose (given that they have compatible cores).

▶ **Lemma 10** (⋆). *Let $\mathcal{A}, \mathcal{B}$ be algorithms satisfying Condition 1 for the same problem Π^κ, $(G, \mathcal{I}, k)$ be an input of Π^κ, $(H, \mathcal{L}, \ell) = \mathcal{B}(G, \mathcal{I}, k)$, and $(\Lambda_\mathcal{B}, \lambda_\mathcal{B})$ be a core for $\mathcal{B}$. If there is some core $(\Lambda_\mathcal{A}, \lambda_\mathcal{A})$ for $\mathcal{A}$ such that $\lambda_\mathcal{A}(\Lambda_\mathcal{A}) = \Lambda_\mathcal{B}$, then the algorithm $\mathcal{C}$ obtained by applying $\mathcal{B}$ and then $\mathcal{A}$ satisfies Condition 1 with core $(\Lambda_\mathcal{A}, \lambda_\mathcal{B} \circ \lambda_\mathcal{A})$.*

▶ **Observation 11.** *If $\mathcal{A}$ is an algorithm that satisfies Condition 1 for core (Λ_H, λ), then for every $\Lambda'_H \subseteq \Lambda_H$ it holds that $\mathcal{A}$ satisfies Condition 1 for the core (Λ'_H, λ'), where λ' is the restriction of λ to Λ'_H.*

As all our kernels are defined by applying exhaustively some reduction rules, let us now explain how we use Lemma 10 and Observation 11 to argue that, once the soundness of each rule is proved, this implies the soundness of the kernel that is defined by the composition of these rules (using an appropriate core).

Let us start with the parameterization by the solution size. In this setting, all reduction rules have the following structure. Given an input (G, t) of the rule, it outputs (G', t'), and we choose as the core the set $V(G')$ with the identity injection. Then, given the input of the kernel (G, t), and the output (H, ℓ) after applying exhaustively all rules, Observation 11 implies, if we choose $V(H)$ as the core (for the whole kernel), that all rules are sound using $V(H)$ as the core, and Lemma 10 implies that the overall kernel is sound as well. Similarly, moving to structural parameterizations, all our rules also have a common structure. Given an input (G, X, t) of the rule, where X is the modulator corresponding to a given parameterization, the rule outputs (G', X', t'), where X' may be a subset or a superset of X, and we always choose as the core the set $X' \cap X$, again with the identity injection. Then, given the input of the kernel (G, X, t), and the output (H, X', ℓ) after applying exhaustively all rules, Observation 11 implies that, if X' is the kernel's core, all rules are sound using X' as the core, and Lemma 10 implies that the overall kernel is sound as well.

4 The natural parameterization for Enum Vertex Cover

Let us formally define our target problem.

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

We use crown decompositions to obtain a simple linear PD kernel for Enum Vertex Cover parameterized by k. While it has a slightly higher constant on the number of vertices than the strong PD kernel presented by Bougeret et al. [7] (3 versus 2), our lifting algorithm takes only a few lines, while theirs is eight pages long. We start with Observation 12, which states that if a compression algorithm only removes vertices, and, in addition, has a "forward safeness proof" that just restricts the solution to the output graph, then we can keep a large core. Recall that having large cores can only make Condition 3 easier to satisfy.

▶ **Observation 12.** *Let $\mathcal{A}_1$ be a* Vertex Cover *kernel that: (i) given (G, k), outputs (G', ℓ) where $G' \coloneqq G \setminus X$ for some $X \subseteq V(G)$; and (ii) for every $S \in \mathsf{Sol}(G, k)$, $S \setminus X \in \mathsf{Sol}(G', \ell)$. Then, by setting $V(G')$ and the identity injection as the core, we satisfy Condition 1.*

We define crowned graphs in Definition 13 and a strengthening of crown decompositions in Definition 14, where the body is small or, equivalently, that the crown is large.

▶ **Definition 13.** *A graph is a* crowned graph *if its vertex set can be partitioned into $H \cup C$, such that C is an independent set, and such that there is a matching M between H and C with $|M| = |H|$. The set C is called the* crown*, and H is called the* head*. Given a graph G, a* crown decomposition *[18, p. 26] of width t is a partition (C, H, B) of $V(G)$ such that $G[H \cup C]$ is a crowned graph (with head H), $N_G(C) = H$, and $|H| = t$; set B is the* body*.*

▶ **Definition 14** (Heavy-crown decomposition)**.** *Given a graph G, a* heavy-crown decomposition *of width t is a crown decomposition (C, H, B) of G of width t where $|B| + 2|H| \leq 3t$.*

We remark that heavy-crown decomposition is in fact the original definition of Chor et al. [14] for crown decompositions. The proof of Lemma 15 is given as the proof of the Crown Lemma in the more recent book by Fomin et al. [27]; while they do not explicitly state it in terms of heavy-crown decompositions, one can easily check that it indeed leads to the conclusion that one can find such a decomposition in polynomial time.

▶ **Lemma 15.** *Let $t \in \mathbb{N}$. There is a polynomial-time algorithm that, given t, and a graph G with no isolated vertices and with $|V(G)| \geq 3t + 1$, either: (i) computes a matching of size at least $t + 1$ of G, or (ii) computes a heavy-crown decomposition of width at most t.*

Let us now explain why we consider the heavy-crown variant instead of the classical one (where B may be arbitrarily large). In the textbook kernel for Vertex Cover, the algorithm finds a crown decomposition, and as there is always at least one solution that contains H, it restarts on $(G', k') = (G \setminus (H \cup C), k - |H|)$. Such a rule verifies the hypothesis of Observation 12, and thus we could use it in a PD kernel with the core $(V(G'), \mathsf{id})$, and Condition 1 would be satisfied. However, proving that Condition 3 holds would become challenging (see how it is done in [7]) as it would require, given a good trace $Y_G \subseteq V(G) \setminus (H \cup C)$, to enumerate with polynomial delay all vertex covers S of G of size at most k such that $S \setminus (H \cup C) = Y_G$, which is non-trivial as $G[H \cup C]$ may be complex (even if C is an independent set). Thus, we prefer to rely on heavy-crown decompositions, which allows us to only remove vertices from C (as in [1]) instead of $H \cup C$ to obtain our kernel.

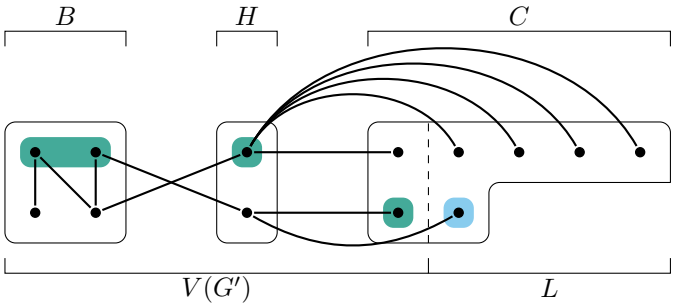

Figure 2 Illustration of the lifting algorithm of Theorem 17, with $k = 5$. The teal-shaded vertices correspond to $Y' \in \mathsf{Sol}(G', k)$, and the cyan-shaded vertex to the mandatory vertex that must be added to Y' to obtain $Y \in \mathsf{Sol}(G, k)$. If we take Y' as the unshaded vertices of G', then we must add the top four vertices of L to Y' to obtain Y, but then $|Y| > k$, and so $Y \notin \mathsf{Sol}(G, k)$.

We consider the kernel for Vertex Cover based on the following two rules, and then show that we can indeed use this kernel to satisfy Conditions 1–3 to get a PD kernel.

▶ **Reduction Rule VC-VC.1.** *Let (G, k) be an input instance of* Vertex Cover. *If v is an isolated vertex of G, remove v from G and set $k \leftarrow k$.*

▶ **Reduction Rule VC-VC.2.** *Let (G, k) be an input of* Vertex Cover *and (C, H, B) be a heavy-crown decomposition of G of width $t \leq k$. Let $M^\star$ be an H-saturating matching between H and C (meaning $|M^\star| = |H|$), and $L = C \setminus V(M^\star)$. Then, output $(G \setminus L, k)$.*

Note that the soundness proofs of Rules VC-VC.1 and VC-VC.2 (choosing as the core the vertex set of the output graph as announced in Section 3) is immediate by Observation 12.

Given an instance (G, k) of Vertex Cover, the compression algorithm $\mathcal{A}_1$ is as follows. First, we apply Rule VC-VC.1 exhaustively. If $|V(G)| \leq 3k$, we output $(G', k') = (G, k)$. Otherwise, apply Rule VC-VC.2 once using the algorithm of Lemma 15 with $t = k$. If it finds a matching of size $k + 1$ then we output a trivial no-instance. Otherwise, output (G', k'), the output of Rule VC-VC.2. Lemma 16 is immediate

▶ **Lemma 16.** *Let (G, k) be an input of* Vertex Cover *and $(G', k) = \mathcal{A}_1(G, k)$. It holds that $|V(G')| \leq 3k$ and $\mathsf{Sol}(G, k) \neq \emptyset$ if and only if $\mathsf{Sol}(G', k') \neq \emptyset$.*

We are now ready to derive our PD kernel using the framework of Section 3.

▶ **Theorem 17.** Enum Vertex Cover *admits a PD kernel with at most $3k$ vertices when parameterized by the maximum size of a solution k.*

Proof. Let us prove how the three conditions of Section 3 can be satisfied. Given an instance (G, k) of Vertex Cover, let (G', k) be the output of $\mathcal{A}_1$. Notice first that, as explained at the end of Section 3, $\mathcal{A}_1$ verifies Condition 1 with core $(V(G'), \mathsf{id})$. Towards verifying Condition 2, let $Y' \in \mathsf{Sol}(G', k)$, $L = V(G) \setminus V(G')$, and observe that, by our reduction rules, L is an independent set. First, we construct $Y = Y' \cup (L \cap N_G(V(G') \setminus Y'))$, that is, we add to Y' the vertices of L that are incident to edges not covered by Y', i.e., they are mandatory in any solution of G that intersects the core at Y' (cf. Figure 2); note that this implies that Y is a vertex cover of G. As such, if $|Y| > k$, we have that Y' is not a good trace. As there is exactly one solution of (G', k') with each good trace and we can decide if a trace is good or not in polynomial time, Condition 2 is satisfied. Finally, Condition 3 is also immediate. Indeed, given a good trace $Y' \subseteq V(G')$, we define Y as above, and take $I = L \setminus Y$. As Y is already a vertex cover, it suffices to enumerate all subsets of I of size at most $k - |Y|$. ◀

References

1. Faisal N. Abu-Khzam. A kernelization algorithm for d-Hitting Set. *Journal of Computer and System Sciences*, 76(7):524–531, 2010. doi:10.1016/j.jcss.2009.09.002.
2. Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. Crown structures for vertex cover kernelization. *Theory of Computing Systems*, 41(3):411–430, 2007. doi:10.1007/s00224-007-1328-0.
3. Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic*, pages 208–222, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-74915-8_18.
4. Valentin Bartier, Oscar Defrain, and Fionn Mc Inerney. Hypergraph dualization with fpt-delay parameterized by the degeneracy and dimension. In *Combinatorial Algorithms: 35th International Workshop, IWOCA 2024, Ischia, Italy, July 1–3, 2024, Proceedings*, pages 111–125, Berlin, Heidelberg, 2024. Springer-Verlag. doi:10.1007/978-3-031-63021-7_9.
5. Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240.
6. J.A. Bondy and U.S.R Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 1st edition, 2008.
7. Marin Bougeret, Guilherme C. M. Gomes, Vinícius Fernandes dos Santos, and Ignasi Sau. Enumeration Kernels for Vertex Cover and Feedback Vertex Set. In *Proc. of the 20th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 358 of *LIPIcs*, pages 23:1–23:18, 2025. The full version is available at https://arxiv.org/pdf/2509.08475. doi:10.4230/LIPIcs.IPEC.2025.23.
8. Marin Bougeret, Bart M. P. Jansen, and Ignasi Sau. Bridge-depth characterizes which minor-closed structural parameterizations of vertex cover admit a polynomial kernel. *SIAM Journal on Discrete Mathematics*, 36(4):2737–2773, 2022. doi:10.1137/21M1400766.
9. Marin Bougeret and Ignasi Sau. How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs? *Algorithmica*, 81(10):4043–4068, 2019. doi:10.1007/s00453-018-0468-8.
10. Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. In Christian Choffrut and Matthias Jantzen, editors, *STACS 91*, pages 348–359, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. doi:10.1007/BFB0020811.
11. Guilherme C. M. Gomes, Emanuel Juliano, Gabriel Martins, and Vinicius F. dos Santos. Matching (Multi)Cut: Algorithms, Complexity, and Enumeration. In *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:15, Dagstuhl, Germany, 2024. doi:10.4230/LIPIcs.IPEC.2024.25.
12. Florent Capelli and Yann Strozecki. Incremental delay enumeration: Space and time. *Discret. Appl. Math.*, 268:179–190, 2019. doi:10.1016/J.DAM.2018.06.038.
13. Nofar Carmeli, Batya Kenig, and Benny Kimelfeld. Efficiently enumerating minimal triangulations. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '17, pages 273–287, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3034786.3056109.
14. Benny Chor, Mike Fellows, and David Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science*, pages 257–269, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30559-0_22.
15. Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

16 Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, and Heribert Vollmer. A complexity theory for hard enumeration problems. *Discrete Applied Mathematics*, 268:191–209, 2019. doi:10.1016/j.dam.2019.02.025.

17 Nadia Creignou, Arne Meier, Julian-Steffen Müller, Johannes Schmidt, and Heribert Vollmer. Paradigms for parameterized enumeration. *Theory of Computing Systems*, 60(4):737–758, May 2017. doi:10.1007/s00224-016-9702-4.

18 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

19 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. On the hardness of losing width. *Theory of Computing Systems*, 54(1):73–82, 2014. doi:10.1007/S00224-013-9480-1.

20 Peter Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006. Parameterized and Exact Computation. doi:10.1016/j.tcs.2005.10.004.

21 Peter Damaschke. Fixed-parameter enumerability of cluster editing and related problems. *Theory of Computing Systems*, 46(2):261–283, 2010. doi:10.1007/s00224-008-9130-1.

22 Rodney G Downey and Michael R Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

23 Arnaud Durand, Nicole Schweikardt, and Luc Segoufin. Enumerating answers to first-order queries over databases of low degree. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '14, pages 121–131, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2594538.2594539.

24 Thomas Eiter, Georg Gottlob, and Kazuhisa Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM Journal on Computing*, 32(2):514–537, 2003. doi:10.1137/S009753970240639X.

25 Henning Fernau. On parameterized enumeration. In Oscar H. Ibarra and Louxin Zhang, editors, *Computing and Combinatorics*, pages 564–573, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-45655-4_60.

26 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

27 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

28 Fedor V. Fomin and Torstein J. F. Strømme. Vertex cover structural parameterization revisited. In *Proc. of the 42nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 9941 of *LNCS*, pages 171–182, 2016. doi:10.1007/978-3-662-53536-3_15.

29 Petr A. Golovach, Christian Komusiewicz, Dieter Kratsch, and Van B. Le. Refined notions of parameterized enumeration kernels with applications to matching cut enumeration. *Journal of Computer and System Sciences*, 123:76–102, 2022. doi:10.1016/j.jcss.2021.07.005.

30 Eva-Maria C. Hols and Stefan Kratsch. Smaller parameters for vertex cover kernelization. In *Proc. of the 12th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 89 of *LIPIcs*, pages 20:1–20:12, 2017. doi:10.4230/LIPIcs.IPEC.2017.20.

31 Bart M. P. Jansen and Hans L. Bodlaender. Vertex Cover Kernelization Revisited - Upper and Lower Bounds for a Refined Parameter. *Theory of Computing Systems*, 53(2):263–299, 2013. doi:10.1007/s00224-012-9393-4.

32 David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988. doi:10.1016/0020-0190(88)90065-8.

33 Benny Kimelfeld and Phokion G. Kolaitis. The complexity of mining maximal frequent subgraphs. *ACM Trans. Database Syst.*, 39(4), December 2015. doi:10.1145/2629550.

34 Christian Komusiewicz and Diptapriyo Majumdar. Enumeration kernels of polynomial size for cuts of bounded degree, 2025. arXiv:2308.01286.

35 Christian Komusiewicz, Diptapriyo Majumdar, and Frank Sommer. Polynomial-size enumeration kernelizations for long path enumeration, 2025. doi:10.48550/arXiv.2502.21164.

36 A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. URL: http://www.jstor.org/stable/1910129.

37 Diptapriyo Majumdar, Venkatesh Raman, and Saket Saurabh. Polynomial kernels for vertex cover parameterized by small degree modulators. *Theory of Computing Systems*, 62(8):1910–1951, 2018. doi:10.1007/s00224-018-9858-1.

38 Kitty Meeks. Randomised enumeration of small witnesses using a decision oracle. *Algorithmica*, 81(2):519–540, 2019. doi:10.1007/s00453-018-0404-y.

39 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, February 2006. doi:10.1093/acprof:oso/9780198566076.001.0001.

40 Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976. doi:10.1016/0304-3975(76)90061-X.

41 Yann Strozecki. *Enumeration complexity and matroid decomposition*. PhD thesis, Paris 7, 2010. URL: http://www.theses.fr/2010PA077178.

42 Yann Strozecki. Enumeration complexity. *Bulletin of EATCS*, 3(129), 2019.