Parameterised Enumeration for Modification Problems

Nadia Creignou, Raïda Ktari, Arne Meier, Julian-Steffen Müller, Frédéric Olive, Heribert Vollmer

MDPI

Article

Parameterised Enumeration for Modification Problems

Nadia Creignou [1], Raïda Ktari [2], Arne Meier [3,*], Julian-Steffen Müller [3], Frédéric Olive [1], Heribert Vollmer [3]

1 Aix Marseille Université, LIS, Marseille, France; nadia.creignou@univ-amu.fr, frederic.olive@lif.univ-mrs.fr
2 Université de Sfax, Pôle technologique de Sfax; raida.ktari@isims.usf.tn
3 Leibniz Universität Hannover, Institut für Theoretische Informatik; meier@thi.uni-hannover.de, vollmer@thi.uni-hannover.de
* Correspondence: meier@thi.uni-hannover.de; Tel.: +49-(0)511-762-19768
† This paper is an extended version of our paper published in the proceedings of Language and Automata Theory and Applications — 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, under the title "Parameterized Enumeration for Modification Problems".

Version August 28, 2019 submitted to Algorithms

Abstract: Recently, Creignou et al. (Theory Comput. Syst. 2017) have introduced the class DelayFPT into parameterised complexity theory in order to capture the notion of efficiently solvable parameterised enumeration problems. In this paper, we propose a framework for parameterised *ordered* enumeration and will show how to obtain enumeration algorithms running with an FPT delay in the context of general modification problems. We study these problems considering two different orders of solutions, namely, lexicographic order and order by size. Furthermore, we present two generic algorithmic strategies. The first one is based on the well-known principle of self-reducibility and is used in the context of lexicographic order. The second one shows that the existence of a neighbourhood structure among the solutions implies the existence of an algorithm running with FPT delay which outputs all solutions ordered non-decreasingly by their size.

Keywords: Parameterised complexity; Enumeration; Bounded Search Tree; Parameterised Enumeration; Ordering

1. Introduction

The study of enumeration problems, that is, the task of generating all solutions of a given computational problem, finds a wealth of applications, e.g., in query answering in databases [1] and web search engines [2], bioinformatics [3] and computational linguistics [4]. From a complexity-theoretic viewpoint, the notion of DelayP, the class of problems whose instance solutions can be output in such a way that the delay between two outputs is bounded by a polynomial, is of high importance [5].

For many enumeration problems often it is central that the output solutions obey a given ordering: in many applications it is interesting to get the solutions with the smallest "cost" at the beginning. Enumerating all solutions in non-decreasing order allows to determine not only the smallest solution, but also the kth-smallest one. Also with such a generating algorithm, it is possible to find the smallest solution satisfying additional constraints in checking at each generation step whether these constraints are satisfied. The disadvantage of this method is that it cannot guarantee fast results because a long prefix of candidates may not satisfy them. However, this technique has the advantage to be applicable to any additional decidable constraint (see, e.g., [6]). Let us illustrate this with some examples.

The question for which classes of propositional CNF formulas enumerating all satisfying solutions is possible in DelayP, as defined above, was studied by Creignou and Hébrard [7]. In terms of the

well-known Schaefer framework for classification of Boolean constraint satisfaction problems, it was shown that for the classes of Horn, anti-Horn, affine or bijunctive formulas, such an algorithm exists. For any other classes of formulas, the existence of a DelayP-algorithm implies P = NP. It is interesting to note that the result hinges on the self-reducibility of the propositional satisfiability problem. Since variables are tried systematically first with an assignment 0 and then 1, it can be observed that the given enumeration algorithms output all satisfying assignments in lexicographic order.

Creignou et al. [8] studied the enumeration of satisfying assignments for propositional formulas under a different order, namely in non-decreasing weight, and it was shown that under this new requirement, enumerating with polynomial delay is only possible for Horn formulas and width-2 affine formulas (i.e., affine formulas with at most 2 literals per clause). One of the main ingredients of these algorithms is the use of a priority queue to ensure enumeration in order (as is observed already by Johnson et al. [5]).

While parameterised enumeration has already been considered before (see, e.g., the works of Fernau, Damaschke and Fomin et al. [9–11]), the notion of fixed-parameter tractable delay was introduced only recently in this context, leading to the definition of the complexity class DelayFPT [12]. The "polynomial time" in the definition of DelayP here is simply replaced by a time-bound of the form $p(n) \cdot f(k)$, where n denotes the input length, k is the input parameter, p is an arbitrary polynomial, and f is a computable function. By this, the notion of efficiency in the context of the parameterised world, i.e., fixed-parameter tractability (FPT), has been combined with the enumeration framework. A number of problems from propositional logic were studied by Creignou et al. [12] and enumeration algorithms based on self-reducibility and on the technique of kernelisation were developed. In particular, it was shown that membership of an enumeration problem in DelayFPT can be characterised by a certain tailored form of kernelisability, very much as in the context of usual decision problems.

As this area of parameterised enumeration is rather young and has received less attention, we want to further push the topic with this paper. Here, we study *ordered enumeration* in the context of *parameterised complexity*. First, we develop a novel formal framework for enumeration with *arbitrary* orders. Then we consider the special context of graph modification problems where we are interested in ordered enumeration for the two mostly studied orders, namely by *lexicographic* and by non-decreasing *size* (where the size is the number of modifications that have to be made). We use two algorithmic strategies, depending on the respective order as follows. Based on the principle of self-reducibility we obtain DelayFPT (and polynomial-space) enumeration algorithms for lexicographic order, as soon as the decision problem is efficiently solvable. Secondly, we present a DelayFPT enumeration algorithm for order by size as soon as a certain FPT-computable neighbourhood function on the solutions set exists (see Theorem 1). Notice that, the presented enumeration algorithms do not start from a minimal solution but solutions of bounded size. Extending to such solutions from minimal ones in the enumeration process is not generally trivial. To cope with the order, we use a priority queue that may require exponential space in the input length (as there exist potentially that many solutions).

Eventually, we show that the observed principles and algorithmic strategies can be applied to general modification problems as well. It is a rather rare situation that a general algorithmic scheme is developed. Usually algorithms are devised on a very individual basis. We prove a wide scope of applicability of our method by presenting new FPT-delay ordered enumeration algorithms for a large variety of problems, such as cluster editing [13], triangulation [14], triangle deletion [15], closest-string [16], and backdoor sets [17]. Furthermore, there already exists work which adopts the introduced framework of Creignou et al. [12] in the area of conjunctive query enumeration [18], triangle enumeration [19], combinatorial optimisation [20], abstract argumentation [21], and global constraints [22].

2. Preliminaries

We start by defining parameterised enumeration problems with a specific ordering and their corresponding enumeration algorithms. Most definitions in this section transfer those of Johnson et al.

and Schmidt [5,23] from the context of enumeration and those of Creignou et al. [12] from the context of parameterised enumeration to the context of parameterised *ordered* enumeration.

The studied orderings of enumeration problems in this paper are quasi-orders which will be defined in the following.

Definition 1 (Quasi-order). *Let R be a set and $\preceq$ a binary relation on R. Then $\preceq$ is a preorder (or quasi-order) if we have for all elements $a, b, c \in R$:*

- *$a \preceq a$, and*
- *if $a \preceq b$ and $b \preceq c$ then $a \preceq c$.*

We will write $z \not\preceq y$ whenever $z \preceq y$ is not true.

Now, we proceed by introducing parameterised enumeration problems with ordering. Intuitively, the corresponding enumeration algorithm for such problems has to obey the given ordering, that is, it has to produce solutions without violating that ordering.

Definition 2. *A* parameterised enumeration problem with ordering *is a quadruple $E = (I, \kappa, \mathrm{Sol}, \preceq)$ such that the following holds:*

- *I is the set of* instances.
- *$\kappa \colon I \to \mathbb{N}$ is the* parameterisation function*; κ is required to be polynomial-time computable.*
- *Sol is a function such that for all $x \in I$, $\mathrm{Sol}(x)$ is a finite set, the set of* solutions *of x. Further we write $\mathcal{S} = \bigcup_{x \in I} \mathrm{Sol}(x)$.*
- *$\preceq$ is a quasi-order on $\mathcal{S}$.*

Notice that this order on all solutions is only a lazy way of simultaneously giving an order for each instance. Furthermore, we will write an index E letter, e.g., I_E, κ_E, to denote that we talk about instance set, parameterisation function, etc. of a given enumeration problem E. In the next step, we fix the notion of enumeration algorithms in our setting.

Definition 3 (Enumeration Algorithm). *Let $E = (I, \kappa, \mathrm{Sol}, \preceq)$ be a parameterised enumeration problem with ordering. Then an algorithm $\mathcal{A}$ is an* enumeration algorithm for *E if the following holds:*

- *For every $x \in I$, $\mathcal{A}(x)$ terminates after a finite number of steps.*
- *For every $x \in I$, $\mathcal{A}(x)$ outputs exactly the elements of $\mathrm{Sol}(x)$ without duplicates.*
- *For every $x \in I$ and $y, z \in \mathrm{Sol}(x)$, if $y \preceq z$ and $z \not\preceq y$ then $\mathcal{A}(x)$ outputs solution y before solution z.*

Before we define complexity classes for parameterised enumeration, we need the notion of delay for enumeration algorithms.

Definition 4 (Delay). *Let $E = (I, \kappa, \mathrm{Sol}, \preceq)$ be a parameterised enumeration problem with ordering and $\mathcal{A}$ be an enumeration algorithm for E. Let $x \in I$ be an instance. The i-th delay of $\mathcal{A}$ is the elapsed runtime with respect to $|x|$ of $\mathcal{A}$ between outputting the i-th and $(i+1)$-st solution in $\mathrm{Sol}(x)$. The 0-th delay is the* precomputation time *which is the elapsed runtime with respect to $|x|$ of $\mathcal{A}$ from the start of the computation to the first output statement. Analogously, the n-th delay, for $n = |\mathrm{Sol}(x)|$, is the* postcomputation time *which is the elapsed runtime with respect to $|x|$ of $\mathcal{A}$ after the last output statement until $\mathcal{A}$ terminates. Then, the* delay of $\mathcal{A}$ *is the maximum over all $0 \leq i \leq n$ of the i-th delay of $\mathcal{A}$.*

Now we are able to define two different complexity classes for parameterised enumeration following the notion of Creignou et al. [12].

Definition 5. *Let $E = (I, \kappa, \mathrm{Sol}, \preceq)$ be a parameterised enumeration problem. We say E is* FPT*-enumerable if there exists an enumeration algorithm $\mathcal{A}$, a computable function $f \colon \mathbb{N} \to \mathbb{N}$, and a polynomial p such that for every $x \in I$, $\mathcal{A}$ outputs all solutions of $\mathrm{Sol}(x)$ in time $f(\kappa(x)) \cdot p(|x|)$.*

An enumeration algorithm $\mathcal{A}$ is a DelayFPT*-algorithm if there exists a computable function $f\colon \mathbb{N} \to \mathbb{N}$, and a polynomial p such that for every $x \in I$, $\mathcal{A}$ outputs all solutions of $\mathrm{Sol}(x)$ with delay of at most $f(\kappa(x)) \cdot p(|x|)$.*

The class DelayFPT *consists of all parameterised enumeration problems that admit a* DelayFPT*-enumeration algorithm.*

Some of our enumeration algorithms will make use of the concept of *priority queues* to enumerate all solutions in the correct order and to avoid duplicates. We will follow the approach of Johnson et al. [5]. For an instance x of a parameterised enumeration problem whose sizes of solutions are polynomially bounded in $|x|$, we use a priority queue Q to store a subset of $\mathrm{Sol}(x)$, of cardinality potentially exponential in $|x|$. The *insert operation* of Q requires $O(|x| \cdot \log |\mathrm{Sol}(x)|)$ time. The *extract minimum operation* requires $O(|x| \cdot \log |\mathrm{Sol}(x)|)$ time, too. It is important, however, that the computation of the order between two elements takes at most $O(|x|)$ time. As pointed out by Johnson et al. the required queue can be implemented with the help of standard balanced tree schemes [24].

2.1. Graph Modification Problems

Graph modifications problems have been studied for a long time in computational complexity theory [25]. Already in the monograph by Garey and Johnson [26], among the graph-theoretic problems considered, many fall into this problem class. To the best of our knowledge, graph modification problems were studied in the context of parameterised complexity for the first time in [27].

In this paper, we consider only undirected graphs. Let $\mathcal{G}$ denote the set of all undirected graphs. A *graph property* $\mathcal{P} \subseteq \mathcal{G}$ is a set of graphs.

Definition 6 (Graph Operations)**.** *Given a graph property $\mathcal{P}$ and an undirected graph G, we write $G \models \mathcal{P}$ if the graph G obeys the property $\mathcal{P}$, that is, $G \in \mathcal{P}$. A* (graph) operation *for G is either of the following:*

- *removing a vertex: a function $\mathrm{rem}_v\colon \mathcal{G} \to \mathcal{G}$ such that $\mathrm{rem}_v(G)$ is the graph obtained by removing the vertex v from G (if v is present; otherwise rem_v is the identity) and deleting all incident edges to v,*
- *adding/removing an edge: a function $\mathrm{add}_{\{u,v\}}, \mathrm{rem}_{\{u,v\}}\colon \mathcal{G} \to \mathcal{G}$ such that $\mathrm{add}_{\{u,v\}}(G)$, $\mathrm{rem}_{\{u,v\}}(G)$ is the graph obtained by adding/removing the edge $\{u,v\}$ to G if u and v are present in G; otherwise both functions are the identity*

Two operations o, o' are dependent *if*

- *$o = \mathrm{rem}_v$ and $o' = \mathrm{rem}_{\{u,v\}}$ (one removes a vertex v and the other removes or adds an edge incident to v), or*
- *$o = \mathrm{rem}_{\{u,v\}}$ and $o' = \mathrm{add}_{\{u,v\}}$ (one removes an edge $\{u,v\}$ and the other adds the same edge $\{u,v\}$ again).*

A set of operations is consistent *if it does not contain two dependent operations. Given such a consistent set of operations S, the graph obtained from G by applying the operations in S on G is denoted by $S(G)$.*

Now, we turn towards the definition of solutions and will define minimality in terms of being inclusion-minimal.

Definition 7 (Solutions)**.** *Given a graph property $\mathcal{P}$, a graph G, $k \in \mathbb{N}$, and a set of operations O, we say that S is a* solution for (G, k, O) with respect to $\mathcal{P}$ *if the following three properties hold:*

1. *$S \subseteq O$ is a consistent set of operations,*
2. *$|S| \leq k$, and*
3. *$S(G) \models \mathcal{P}$.*

A solution S is minimal *if there is no solution S' such that $S' \subsetneq S$.*

Cai [27] was interested in the following parameterised graph modification decision problem with respect to a given graph property $\mathcal{P}$:

Problem:	$\mathcal{M}_\mathcal{P}$
Input:	(G, k, O), G undirected graph, $k \in \mathbb{N}$, O set of operations on G.
Parameter:	The integer k.
Question:	Does there exist a solution for (G, k, O) with respect to $\mathcal{P}$?

Some of the most important examples of graph modification problems are presented now. A *chord* in a graph $G = (V, E)$ is an edge between two vertices of a cycle C in G which is not part of C. A given graph $G = (V, E)$ is *triangular* (or *chordal*) if each of its induced cycles of 4 or more nodes has a chord. The problem TRIANGULATION then asks, given an undirected graph G and $k \in \mathbb{N}$, whether there exists a set of at most k edges such that adding this set of edges to G makes it triangular. Yannakakis showed that this problem is NP-complete [14]. Kaplan et al. [28], and independently Cai [27] have shown that the parameterised problem is in FPT. For this problem, a solution is a set of edges which have to be added to the graph to make the graph triangular. Observe that, in this special case of the modification problem, the underlying property $\mathcal{P}$, "to be triangular", does not have a finite forbidden set characterisation (since cycles of any length are problematic). Nevertheless, we will see later, that one can efficiently enumerate all minimal solutions as well.

A *cluster* is a graph such that all its connected components are cliques. In order to transform (or modify) a graph G we allow here only two kinds of operations: adding or removing an edge. CLUSTER-EDITING asks, given a graph G and a parameter k, whether there exists a consistent set of operations of cardinality at most k such that $S(G)$ is cluster. It was shown by Shamir et al. that the problem is NP-complete [13].

The problem TRIANGLE-DELETION asks whether a given graph can be transformed into a triangle-free graph by deletion of at most k vertices. Yannakakis has shown that the problem is NP-complete [15].

Analogous problems can be defined for many other classes of graphs, e.g., line graphs, claw-free graphs, Helly circular-arc graphs, etc., see [29].

Now, we turn towards the main focus of the paper. Here, we are interested in corresponding enumeration problems with ordering. In particular, we will focus on two well-known preorders, lexicographic ordering and ordering by size. Since our solutions are subsets of an ordered set of operations, they can be encoded as binary strings in which the ith bit from right indicates whether the ith operation is in the subset. We define the lexicographic ordering of solutions as the lexicographic ordering of these strings. Then, the size of a solution simply is its cardinality.

Problem:	ENUM-$\mathcal{M}_\mathcal{P}^{\text{LEX}}$
Input:	(G, k, O), G undirected graph, $k \in \mathbb{N}$, O ordered set of operations on G.
Parameter:	The integer k.
Output:	All solutions of (G, k, O) with respect to $\mathcal{P}$ in lexicographic order.

Problem:	ENUM-$\mathcal{M}_\mathcal{P}^{\text{SIZE}}$
Input:	(G, k, O), G undirected graph, $k \in \mathbb{N}$, O set of operations on G.
Parameter:	The integer k.
Output:	All solutions of (G, k, O) with respect to $\mathcal{P}$ in non-decreasing size.

If the context is clear, we omit the subscript $\mathcal{P}$ for the graph modification problem and simply write $\mathcal{M}$. Furthermore, we write $\text{Sol}_\mathcal{M}(x)$ for the function associating solutions to a given instance, and also $\mathcal{S}_\mathcal{M}$ for the set of all solutions of $\mathcal{M}$.

3. Enumeration of Graph Modification Problems with Ordering

In this section, we study the two previously introduced parameterised enumeration problems with ordering (lexicographic and size ordering).

3.1. Lexicographic Ordering

We first prove that, for any graph property $\mathcal{P}$, if the decision problem $\mathcal{M}_\mathcal{P}$ is in FPT then there is an efficient enumeration algorithm for ENUM-$\mathcal{M}_\mathcal{P}^{\text{LEX}}$.

Lemma 1. *Let $\mathcal{M}_\mathcal{P}$ be a graph modification problem. If $\mathcal{M}_\mathcal{P}$ is in* FPT *then* ENUM-$\mathcal{M}_\mathcal{P}^{\text{LEX}} \in$ DelayFPT *with polynomial space.*

Proof. Algorithm 1 enumerates all solutions of an instance of a given modification problem $\mathcal{M}_\mathcal{P}$ by the method of self-reducibility (it is an extension of the flash light search of Creignou and Hébrard [7]). The algorithm uses a function $\texttt{ExistsSol}(G, k, O)$ that tests if the instance (G, k, O) of the modification problem $\mathcal{M}_\mathcal{P}$ has a solution. By assumption of the lemma, $\mathcal{M}_\mathcal{P} \in$ FPT so this function runs in fpt-time. We use calls to this function to avoid exploration of branches of the recursion tree that do not lead to any output. Also, we ensure that the solutions using o_p have to be consistent. This consistency check runs in polynomial time for graph operations. The rest yields a search tree of depth at most k. From this it follows that, for any instance of length n, the time beween the output of any two solutions is bounded by $f(k) \cdot p(n)$ for some polynomial p and a computable function f. □

Algorithm 1: Enumerate all solutions of $\mathcal{M}_\mathcal{P}$ in lexicographic order

Input: (G, k, O): a graph G, $k \in \mathbb{N}$, an ordered set of operations $O = \{o_1, \ldots, o_n\}$
Output: all consistent sets $S \subseteq O$ s.t. $|S| \leq k$, $S(G) \models \mathcal{P}$ in lexicographic order
1 **if** $\texttt{ExistsSol}(G, k, O)$ **then** $\texttt{Generate}(G, k, O, \emptyset)$;

Procedure $\texttt{Generate}(G, k, O, S)$:
1 **if** $O = \emptyset$ *or* $k = 0$ **then return** S;
2 **else**
3 let o_p be the lexicographically last operation in O, $O := O \setminus \{o_p\}$;
4 **if** $\texttt{ExistsSol}(S(G), k, O)$ **then** $\texttt{Generate}(S(G), k, O, S)$;
5 **if** $S \cup \{o_p\}$ *is consistent and* $\texttt{ExistsSol}((S \cup \{o_p\})(G), k-1, O)$ **then**
6 $\texttt{Generate}((S \cup \{o_p\})(G), k-1, O, S \cup \{o_p\})$.

Corollary 1. ENUM-TRIANGULATION$^{\text{LEX}} \in$ DelayFPT *with polynomial space.*

Proof. Kaplan et al. [28] and Cai [27] showed that TRIANGULATION $\in$ FPT. Now, by applying Lemma 1, we get the result. □

Cai [27] identified a class of graph properties whose associated modification problems belong to FPT. Let us introduce some terminology.

Definition 8. *Given two graphs $G = (V, E)$ and $H = (V', E')$, we write $H \trianglelefteq G$ if H is an induced subgraph of G, i.e., $V' \subseteq V$ and $E' = E \cap (V' \times V')$. Let $\mathcal{F}$ be a set of graphs and $\mathcal{P}$ be a graph property. We say that $\mathcal{F}$* is a forbidden set characterisation *of $\mathcal{P}$ if for any graph G it holds that: $G \models \mathcal{P}$ iff for all $H \in \mathcal{F}$, $H \not\trianglelefteq G$.*

Among the problems presented in the previous section (see page 5), TRIANGLE-DELETION and CLUSTER-EDITING have a finite forbidden set characterisation, namely by triangles and paths of length two. In contrast to that, TRIANGULATION has a forbidden set characterisation which is infinite,

since cycles of arbitrary length are problematic. Actually, for properties having a finite forbidden set characterisation, the corresponding modification problem is fixed-parameter tractable. Together with Lemma 1, this provides a positive result in terms of enumeration.

Proposition 1 ([27]). *If a property $\mathcal{P}$ has a* finite *forbidden set characterisation then $\mathcal{M}_\mathcal{P}$ is in* FPT.

Corollary 2. *For any graph modification problem, if $\mathcal{P}$ has a finite forbidden set characterisation then* ENUM-$\mathcal{M}_\mathcal{P}^{\text{LEX}} \in$ DelayFPT *with polynomial space.*

Proof. This result follows by combining Proposition 1 with Lemma 1. □

3.2. Size Ordering

A common strategy in the enumeration context consists of defining a notion of a neighbourhood that allows to compute a new solution from a previous one with small amounts of computation time (see, e.g., the work of Avis and Fukuda [30]). We introduce the notion of a neighbourhood function, which, roughly speaking, generates some initial solutions from which all solutions can be produced. A priority queue then takes care of the ordering and avoids duplicates, which may require exponential space. For the graph modification problems of interest, we show that if the inclusion-minimal solutions can be generated in FPT, then such a neighbourhood function exists, accordingly providing a DelayFPT-enumeration algorithm. In the following, $\mathbb{O}$ (the "seed") is a technical symbol that will be used to generate the initial solutions.

Definition 9. *Let $\mathcal{M}$ be a graph modification problem. A* neighbourhood function *for $\mathcal{M}$ is a (partial) function $\mathcal{N}_\mathcal{M} \colon I_\mathcal{M} \times (\mathcal{S}_\mathcal{M} \cup \{\mathbb{O}\}) \to 2^{\mathcal{S}_\mathcal{M}}$ such that the following holds:*

1. *For all $x = (G, k, O) \in I_\mathcal{M}$ and $S \in \text{Sol}_\mathcal{M}(x) \cup \{\mathbb{O}\}$, $\mathcal{N}_\mathcal{M}(x, S)$ is defined.*
2. *For all $x \in I_\mathcal{M}$, $\mathcal{N}_\mathcal{M}(x, \mathbb{O}) = \emptyset$ if $\text{Sol}_\mathcal{M}(x) = \emptyset$, and $\mathcal{N}_\mathcal{M}(x, \mathbb{O})$ is an arbitrary set of solutions otherwise.*
3. *For all $x \in I_\mathcal{M}$ and $S \in \text{Sol}_\mathcal{M}(x)$, if $S' \in \mathcal{N}_\mathcal{M}(x, S)$ then $|S| < |S'|$.*
4. *For all $x \in I_\mathcal{M}$ and all $S \in \text{Sol}_\mathcal{M}(x)$, there exists $p > 0$ and $S_1, \ldots, S_p \in \text{Sol}_\mathcal{M}(x)$ such that (i) $S_1 \in \mathcal{N}_\mathcal{M}(x, \mathbb{O})$, (ii) $S_{i+1} \in \mathcal{N}_\mathcal{M}(x, S_i)$ for $1 \leq i < p$, and (iii) $S_p = S$.*

Furthermore, we say that $\mathcal{N}_\mathcal{M}$ is FPT*-computable, when $\mathcal{N}_\mathcal{M}(x, S)$ is computable in time $f(\kappa(x)) \cdot poly(|x|)$ for any $x \in I_\mathcal{M}$ and $S \in \text{Sol}_\mathcal{M}(x)$.*

As a result, a neighbourhood function for a problem $\mathcal{M}$ is a function that in a first phase computes from scratch an initial set of solutions (see Definition 9(2)). In many of our applications below, $\mathcal{N}_\mathcal{M}(x, \mathbb{O})$ will be the set of all minimal solutions for x. In a second phase these solutions are iteratively extended (see condition (3)), where condition (4) guarantees that we do not miss any solution, as we will see in the next theorem.

Theorem 1. *Let $\mathcal{M}$ be a graph modification problem. If $\mathcal{M}$ admits a neighbourhood function $\mathcal{N}_\mathcal{M}$ that is* FPT*-computable, then* ENUM -$\mathcal{M}^{\text{SIZE}} \in$ DelayFPT.

Proof. Algorithm 2 outputs all solutions in DelayFPT-time. By the definition of the priority queue (recall in particular that insertion of an element is done only if the element is not yet present in the queue) and by the fact that all elements of $\mathcal{N}_\mathcal{M}((G, k, O), S)$ are of bigger size than S by Definition 9(3), it is easily seen that *the solutions are output in the right order* and that *no solution is output twice*.

Besides, *no solution is omitted*. Indeed, given $S \in \text{Sol}_\mathcal{M}(G, k, O)$ and $S_1, \ldots, S_p$ associated with S by Definition 9(4), we prove by induction that each S_i is inserted in Q during the run of the algorithm:

$i = 1$**:** This proceeds from line 2 of the algorithm.

Algorithm 2: DelayFPT algorithm for ENUM-$\mathcal{M}$

Input : (G, k, O) : G is an undirected graph, $k \in \mathbb{N}$, and O is a set of operations.

1 compute $\mathcal{N}_\mathcal{M}((G, k, O), \mathbb{O})$;
2 insert all elements of $\mathcal{N}_\mathcal{M}((G, k, O), \mathbb{O})$ into priority queue Q (ordered by size);
3 **while** Q *is not empty* **do**
4 **extract** the minimum solution S of Q and output it;
5 **insert** all elements of $\mathcal{N}_\mathcal{M}((G, k, O), S)$ into Q;

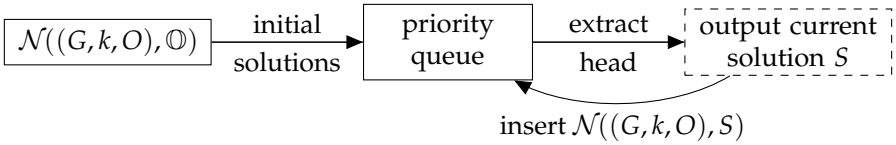

Figure 1. Structure of Algorithm 2.

$i > 1$**:** The solution S_{i-1} is inserted in Q by induction hypothesis and hence all elements of $\mathcal{N}_\mathcal{M}((G, k, O), S_{i-1})$, including S_i, are inserted in Q (line 5 of Algorithm 2). Consequently, each S_i is inserted in Q and then output during the run. In particular, this is true for $S = S_p$.

Finally, we claim that Algorithm 2 runs in DelayFPT-time. Indeed, the delay between the output of two consecutive solutions is bounded by the time required to compute a neighbourhood of the form $\mathcal{N}_\mathcal{M}((G, k, O), \mathbb{O})$ or $\mathcal{N}_\mathcal{M}((G, k, O), S)$ and to insert all its elements in the priority queue. This is in FPT due to the assumption on $\mathcal{N}_\mathcal{M}$ being FPT-computable and as there is only a single extraction and FPT-many insertion operations on the queue. □

A natural way to provide a neighbourhood function for a graph modification problem $\mathcal{M}$ is to consider the *inclusion minimal* solutions of $\mathcal{M}$. Let us denote by MIN-$\mathcal{M}$ the problem of enumerating all inclusion minimal solutions of $\mathcal{M}$.

Theorem 2. *Let $\mathcal{M}$ be a graph modification problem. If* MIN-$\mathcal{M}$ *is* FPT*-enumerable then* ENUM -$\mathcal{M}^{\text{SIZE}} \in$ DelayFPT.

Proof. Let $\mathcal{A}$ be an FPT-algorithm for MIN-$\mathcal{M}$. Because of Theorem 1, it is sufficient to build an FPT-neighbourhood function for $\mathcal{M}$. For an instance (G, k, O) of $\mathcal{M}$ and for $S \in \text{Sol}_\mathcal{M}(G, k, O) \cup \{\mathbb{O}\}$, we define $\mathcal{N}_\mathcal{M}((G, k, O), S)$ as the result of Algorithm 3.

Accordingly, the function $\mathcal{N}_\mathcal{M}$ clearly fulfils Conditions 2 and 3 of Definition 9. We prove by induction that it also satisfies Condition 4 (that is, each solution T of size k comes with a sequence $T_1, \ldots, T_p = T$ such that $T_1 \in \mathcal{N}_\mathcal{M}((G, k, O), \mathbb{O})$ and $T_{i+1} \in \mathcal{N}_\mathcal{M}((G, k, O), T_i)$ for each i). If T is a minimal solution for (G, k, O), then $T \in \mathcal{N}_\mathcal{M}((G, k, O), \mathbb{O})$ and the expected sequence $(T_i)_{1 \leq i \leq p}$ reduces to $T_1 = T$. Otherwise, there exists an $S \in \text{Sol}_\mathcal{M}(G, k, O)$ and a non-empty set of transformations, say $S' \cup \{t\}$, such that $T = S \cup S' \cup \{t\}$ and there is no solution for G between S and $S \cup S' \cup \{t\}$. This entails that S' is a minimal solution for $\big((S \cup \{t\})(G), k - |S| - 1\big)$ and, as a consequence, $T \in \mathcal{N}_\mathcal{M}((G, k, O), S)$ (see lines 4–5 of Algorithm 3). The conclusion follows from the induction hypothesis that guarantees the existence of solutions $S_1, \ldots, S_q$ such that $S_1 \in \mathcal{N}_\mathcal{M}((G, k, O), \mathbb{O})$, $S_{i+1} \in \mathcal{N}_\mathcal{M}((G, k, O), S_i)$ and $S_q = S$. The expected sequence $T_1, \ldots, T_p$ for T is nothing but $S_1, \ldots, S_q, T$. To conclude, it remains to show that Algorithm 3 is FPT. This follows from the fact that $\mathcal{A}$ is an FPT-algorithm (Lines 1 and 4 of Algorithm 3). □

Corollary 3. ENUM-TRIANGULATION$^{\text{SIZE}} \in$ DelayFPT.

Algorithm 3: Procedure for computing $\mathcal{N}_{\mathcal{M}}((G,k,O),S)$

Input : $(G,k,O), S$: G is an undirected graph, $k \in \mathbb{N}$, O and S are sets of operations.

```
if S = 𝕆 then return A(G,k,O);
res := ∅;
forall the t ∈ O do
    forall the S' ∈ A((S ∪ {t})(G), k − |S| − 1, O \ {t}) do
        if S ∪ S' ∪ {t} is consistent then res := res ∪ {S ∪ S' ∪ {t}} ;
return res;
```

Proof. All size minimal k-triangulations can be output in time $O(2^{4k} \cdot |E|)$ for a given graph G and $k \in \mathbb{N}$ as shown by Kaplan et al. [28, Thm. 2.4]. This immediately yields the expected result via Theorem 2. □

Corollary 4. *For any property $\mathcal{P}$ that has a finite forbidden set characterisation, the problem* Enum-$\mathcal{M}_{\mathcal{P}}^{\text{SIZE}}$ *is in* DelayFPT.

Proof. The algorithm developed by Cai [27] for the decision problem is based on a bounded search tree, whose exhaustive examination provides all size minimal solutions in FPT. Theorem 2 yields the conclusion. □

Corollary 5. Enum-Cluster-Editing$^{\text{SIZE}}$ *and* Enum-Triangle-Deletion$^{\text{SIZE}}$ *are in* DelayFPT.

Proof. Both problems have a finite forbidden set characterisation. For the cluster editing problem, paths of length two are the forbidden pattern, and, Regarding Enum-Triangle-Deletion$^{\text{SIZE}}$, the forbidden patterns are obviously just triangles. Finally, just apply Corollary 4. □

4. Generalisation to Modification Problems

In this section, we will show how the algorithmic strategy that has been defined and formalised in the context of graph modification can be of use for many other problems, coming from various combinatorial frameworks.

Definition 10 (General Operations). *Let $Q \subseteq \Sigma^*$ be a language defined over an alphabet, and $x \in \Sigma^*$ be an input. A* set of operations $\Omega(Q) = \{\, \omega_n : \Sigma^* \to \Sigma^* \mid n \in \mathbb{N} \,\}$ *is an infinite set of operations on instances of Q. We say an operation ω is* valid *with respect to an instance $x \in Q$, if $\omega(x) \in Q$. We write Ω/x as the set of possible (valid) operations on an instance x.*

Two operations ω, ω' are dependent *with respect to an instance $x \in Q$ if*

- $\omega(\omega'(x)) = x$, *or*
- $\omega(\omega'(x)) = \omega'(x)$ *or* $\omega(\omega'(x)) = \omega(x)$

A set of operations $O \subseteq \Omega/x$ is consistent *with respect to x if it does not contain two dependent operations.*

For instance, the set Ω could contain operations that add edges or, in another case, flip bits. It highly is the subject to the repective language Q.

Example 1. *Let $\mathcal{G} \subseteq \{0,1\}^*$ be the language of all undirected graphs encoded by adjacency matrices. Then $\Omega(\mathcal{G})$ is the set of all graph operations in the sense of Definition 6: removing vertices or edges, adding edges. Note that $\Omega(\mathcal{G})$ contains all operations of the kind*

$$\text{rem}_i : \mathcal{G} \to \mathcal{G}, \quad \text{rem}_{\{i,j\}} : \mathcal{G} \to \mathcal{G}, \quad \text{add}_{\{i,j\}} : \mathcal{G} \to \mathcal{G}$$

for all $i, j \in \mathbb{N}$. Furthermore, let $G = (V, E) \in \{0,1\}^*$ be a concrete input graph. As a result, Ω/G then is the restriction of Ω to those $i, j \in \mathbb{N}$ such that $v_i, v_j \in V$ encode vertices in G.

Similarly as defined in Subsection 2.1, a property is just a set. In the following context, it is a subset of a considered language Q. Intuitively, you may think, in the view of graph modification problems, of Q as $\mathcal{G}$. Then a graph property $\mathcal{P}$ was just a subset of $\mathcal{G}$.

Definition 11 (General Solutions)**.** *Let $Q \subseteq \Sigma^*$ be a language defined over an alphabet, $S \subseteq \Omega/x$ be a finite set of operations on $x \in Q$ and $\mathcal{P} \subseteq Q$ be a property. We say S is a* solution (of x) *if there exists an ordering of $S = \{\omega_{i_1}, \ldots, \omega_{i_k}\}$ such that $\omega_{i_1}(\omega_{i_2}(\cdots \omega_{i_k}(x) \cdots)) \in \mathcal{P}$ for $i_\mu \in \mathbb{N}$ and $1 \leq \mu \leq k$. In such a case, we also just write $\omega_{i_1}(\omega_{i_2}(\cdots \omega_{i_k}(x) \cdots)) \models \mathcal{P}$. If for every pair of permutations on k elements α, β we have that*

$$\omega_{\alpha(1)}(\omega_{\alpha(2)}(\cdots \omega_{\alpha(k)}(x) \cdots)) = \omega_{\beta(1)}(\omega_{\beta(2)}(\cdots \omega_{\beta(k)}(x) \cdots)),$$

then we say S is consistent*.*

If S is a consistent set of operations then we write $S(x)$ for the application of the operations in S to x. In short, whenever S is a consistent solution we just write $S(x) \models \mathcal{P}$. Similarly, we say an operation ω is consistent with a set *S if and only if $S \cup \{\omega\}$ is consistent. Furthermore, we denote by $\mathcal{S}_Q := \bigcup_{x \in Q} \{\, S \mid S \text{ is a solution of } x \,\}$ the set of all solutions for every instance $x \in Q$. Also $\mathrm{Sol}(x)$ is the set of solutions for every instance $x \in Q$.*

Example 2. *Continuing the previous example, if the property $\mathcal{P}$ is "to be a cluster" then a consistent solution S to a given graph just then is a sequence of removing vertices, adding and deleting of edges where*

- *there is no edge (i, j) added or deleted such that vertex i or j is removed,*
- *there is no edge (i, j) added and removed, and*
- *$S(G) \models \mathcal{P}$.*

Similarly, adding edge (i, j) together with removing vertex i or j or removing edge (i, j) is an inconsistent set of operations.

Now we want to define the corresponding decision and enumeration tasks. On that account, let $\mathcal{P}$ be a property, $\Pi = (Q, \kappa)$ be a parametrised problem with $Q \subseteq \Sigma^*$, and Ω be a set of operations.

Problem:	$\Pi_\mathcal{P}$ — parameterised modification problem Π regarding property $\mathcal{P}$ over Σ
Input:	$x \in \Sigma^*$, $k \in \mathbb{N}$, Ω/x set of operations.
Parameter:	The integer k.
Question:	Is there a consistent solution $S \subseteq \Omega/x$ such that $S(x) \models \mathcal{P}$ and $\lvert S \rvert \leq k$?

Problem:	ENUM-MIN-$\Pi_\mathcal{P}$ — parameterised minimum enumeration modification problem regarding property $\mathcal{P}$ over Σ
Input:	$x \in \Sigma^*$, $k \in \mathbb{N}$, Ω/x set of operations.
Parameter:	The integer k.
Output:	All minimal (w.r.t. some order) consistent solutions $S \subseteq \Omega/x$ with $\lvert S \rvert \leq k$ such that $S(x) \models \mathcal{P}$.

The enumeration modification problem where we want to output all possible sets of transformations on a given instance x (and not only the minimum ones) then is ENUM-$\Pi_\mathcal{P}$.

In the following, we show how the notion of neighbourhood functions can be generalised as well. This will in turn yield generalisations of the results for graph modification problems afterwards.

Definition 12. *Let Σ be an alphabet, $\mathcal{P} \subseteq \Sigma^*$ be a property and $\Pi_\mathcal{P}$ be a parameterised modification problem over Σ. A* neighbourhood function *for $\Pi_\mathcal{P}$ is a (partial) function $\mathcal{N}_{\Pi_\mathcal{P}} \colon \Sigma^* \times \left(\mathcal{S}_{\Pi_\mathcal{P}} \cup \{\mathbb{O}\}\right) \to 2^{\mathcal{S}_{\Pi_\mathcal{P}}}$ such that the following holds:*

1. *For all* $x \in \Sigma^*$ *and* $S \in \mathrm{Sol}_{\Pi_\mathcal{P}}(x) \cup \{\mathbb{O}\}$, $\mathcal{N}_{\Pi_\mathcal{P}}(x,S)$ *is defined.*
2. *For all* $x \in \Sigma^*$, $\mathcal{N}_{\Pi_\mathcal{P}}(x,\mathbb{O}) = \emptyset$ *if* $\mathrm{Sol}_{\Pi_\mathcal{P}}(x) = \emptyset$, *and* $\mathcal{N}_{\Pi_\mathcal{P}}(x,\mathbb{O})$ *is an arbitrary set of solutions otherwise.*
3. *For all* $x \in \Sigma^*$ *and* $S \in \mathrm{Sol}_{\Pi_\mathcal{P}}(x)$, *if* $S' \in \mathcal{N}_{\Pi_\mathcal{P}}(x,S)$ *then* $|S| < |S'|$.
4. *For all* $x \in \Sigma^*$ *and all* $S \in \mathrm{Sol}_{\Pi_\mathcal{P}}(x)$, *there exists* $p > 0$ *and* $S_1, \ldots, S_p \in \mathrm{Sol}_{\Pi_\mathcal{P}}(x)$ *such that (i)* $S_1 \in \mathcal{N}_{\Pi_\mathcal{P}}(x,\mathbb{O})$, *(ii)* $S_{i+1} \in \mathcal{N}_{\Pi_\mathcal{P}}(x,S_i)$ *for* $1 \leq i < p$, *and (iii)* $S_p = S$.

Furthermore, we say that $\mathcal{N}_{\Pi_\mathcal{P}}$ *is* FPT*-computable, when* $\mathcal{N}_{\Pi_\mathcal{P}}(x,S)$ *is computable in time* $f(k) \cdot poly(|x|)$ *for any* $x \in \Sigma^*$ *and* $S \in \mathrm{Sol}_{\Pi_\mathcal{P}}(x)$.

As already announced before, we are able to state generalised versions of Theorems 1 and 2 which can be proven in a similar way. However, one has to replace the graph modification problems by general modification problems.

Corollary 6. *Let* $\mathcal{P}$ *be a property,* $\Pi \subseteq \Sigma^* \times \mathbb{N}$ *be a parameterised modification problem, and* Ω *be a set of operations such that* Ω/x *is finite for all* $x \in \Sigma^*$. *If* $\Pi_\mathcal{P}$ *admits a neighbourhood function that is* FPT*-computable then* ENUM-$\Pi_\mathcal{P} \in$ DelayFPT *and*

- *polynomial space for lexicographic order, and*
- *exponential space for size order.*

Corollary 7. *Let* $\mathcal{P}$ *be a property,* $\Pi \subseteq \Sigma^* \times \mathbb{N}$ *be a parameterised modification problem, and* Ω *be a set of operations such that* Ω/x *is finite for all* $x \in \Sigma^*$. *If* ENUM-MIN-$\Pi_\mathcal{P}$ *is* FPT*-enumerable and consistency of solutions can be checked in* FPT *then* ENUM-$\Pi_\mathcal{P} \in$ DelayFPT *and*

- *polynomial space for lexicographic order, and*
- *exponential space for size order.*

4.1. Closest String

In the following, we consider a central NP-complete problem in coding theory [31]. Given a set of binary strings I, we want to find a string s whose maximum Hamming distance $\max\{\, d_H(s,s') \mid s' \in I \,\} \leq d$ for a $d \in \mathbb{N}$, where $d_H(s,s')$ is the Hamming distance between two strings.

Definition 13 (Bit-flip operation)**.** *Given a string* $w = w_1 \cdots w_n$ *with* $w_i \in \{0,1\}, n \in \mathbb{N}$, *and a set* $S \subseteq \{1,\ldots,n\}$, $S(w)$ *denotes the string obtained from* w *in flipping the bits indicated by* S, *more formally* $S(w) := S(w_1) \cdots S(w_n)$, *where* $S(w_i) = 1 - w_i$ *if* $i \in S$ *and* $S(w_i) = w_i$ *otherwise.*

The corresponding parametrised version is the following.

Problem:	CLOSEST-STRING
Input:	A sequence $(s_1, s_2, ..., s_k)$ of k strings over $\{0,1\}$ each of given length $n \in N$ and an integer $d \in N$..
Parameter:	The integer d.
Question:	Does there exist $S \subseteq \{1,\ldots,n\}$ such that $d_H(S(s_1), s_i) \leq d$ for all $1 \leq i \leq k$?

Proposition 2 ([16])**.** CLOSEST-STRING *is in* FPT.

Moreover, an exhaustive examination of a bounded search tree constructed from the idea of Gramm et al. [16, Fig. 1] allows to produce all minimal solutions of this problem in FPT. Accordingly, we get the following result for the corresponding enumeration problems.

Theorem 3.

- ENUM-CLOSEST-STRING$^{\text{LEX}} \in$ DelayFPT *with polynomial space.*

- ENUM-CLOSEST-STRING$^{\text{SIZE}} \in$ DelayFPT *with exponential space.*

Proof. Ω is just the set of operations which flip the i-th bit of a string for every $i \in \mathbb{N}$. Then use Proposition 2 and Corollary 7. □

4.2. Backdoors

In this section, we will consider the concept of backdoors. Let $\mathcal{C}$ be a class of propositional formulas. Intuitively, a $\mathcal{C}$-backdoor is a set of variables of a given propositional formula with the following property. Applying assignments over these variables to the formula always yields a formula in the class $\mathcal{C}$. Of course, one aims for formula classes for which satisfiability can be decided efficiently. Informally speaking, with the parameter backdoor size of a formula one tries to describe a distance to tractability. This definition was first introduced by Golmes, Williams and Selman [17] to model short distances to efficient subclasses. Until today, backdoors gained copious attention in many different areas: abduction [32], answer set programming [33,34], argumentation [35], default logic [36], temporal logic [37], planning [38], and constraint satisfaction [39,40].

Consider a formula ϕ in conjunctive normal form. Denote by $\phi[\tau]$ for a partial truth assignment τ the result of removing all clauses from ϕ which contain a literal ℓ with $\tau(\ell) = 1$ and removing literals ℓ with $\tau(\ell) = 0$ from the remaining clauses.

Definition 14. *Let $\mathcal{C}$ be a class of CNF-formulas and ϕ be a CNF-formula. A set $V \subseteq Vars(\phi)$ of variables of ϕ is a* strong $\mathcal{C}$-backdoor set *of ϕ if for all truth assignments $\tau\colon V \to \{0,1\}$ we have that $\phi[\tau] \in \mathcal{C}$.*

Definition 15 ([41,42])**.** *Let $\mathcal{C}$ be a class of CNF-formulas and ϕ be a CNF-formula. A set $V \subseteq Vars(\phi)$ of variables of ϕ is a* $\mathcal{C}$-deletion backdoor set *of ϕ if $\phi[V]$ is in $\mathcal{C}$, where $\phi[V]$ denotes the formula obtained from ϕ in deleting in ϕ all occurrences of variables from V.*

Definition 16 (Weak Backdoor Sets)**.** *Let $\mathcal{C}$ be a class of CNF-formulas, and ϕ be a propositional CNF formula. A set $V \subseteq Vars(\phi)$ of variables from ϕ is a* weak $\mathcal{C}$-backdoor set of ϕ *if there exists an assignment $\theta \in \Theta(V)$ such that $\phi[\theta] \in \mathcal{C}$ and $\phi[\theta]$ is satisfiable.*

Now let us consider the following enumeration problem.

Problem:	ENUM-WEAK-BACKDOORSET($\mathcal{C}$)
Input:	A formula ϕ in CNF, $k \in \mathbb{N}$.
Parameter:	The integer k.
Output:	The set of all weak $\mathcal{C}$-backdoor sets of ϕ of size at most k.

Similarly, define ENUM-STRONG-BACKDOORSET($\mathcal{C}$) as the set of all strong $\mathcal{C}$-backdoor sets of ϕ of size at most k. Observe that the backdoor set problems can be seen as modification problems where solutions are sequences of variable assignments. The target property then simply is the class of CNF formulas $\mathcal{C}$.

Notice that Creignou et al. [12, Thm. 4] have studied enumeration for exact strong HORN-backdoor sets and provided an algorithm running in DelayFPT, where HORN denotes the set of all Horn-formulas, that is, CNF-formulas whose clauses contain at most one positive literal.

Definition 17 (Base Class, [43])**.** *The class $\mathcal{C}$ is a* base class *if it can be recognised in* P *(that is, $\mathcal{C} \in$ P), satisfiability of its formulas is in* P*, and the class is closed under isomorphisms w.r.t. variable names. We say $\mathcal{C}$ is* clause-defined *if for every CNF-formula ϕ we have: $\phi \in \mathcal{C}$ if and only if $\{C\} \in \mathcal{C}$ for all clauses C from ϕ.*

Proposition 3 ([43, Prop. 2])**.** *For every clause-defined base class $\mathcal{C}$, detection of weak $\mathcal{C}$-backdoor sets is in* FPT *for input formulas in* 3*CNF.*

In their proof, Gaspers and Szeider [43] describe how utilising a bounded search tree allows one to solve the detection of weak $\mathcal{C}$-backdoors in FPT time. Interestingly to note, this technique results in obtaining *all minimal solutions* in FPT-time. This observation results in the following theorem.

Theorem 4. *For every clause-defined base class $\mathcal{C}$ and input formulas in 3-CNF*

- ENUM-WEAK-$\mathcal{C}$-BACKDOORS$^{\text{LEX}} \in$ DelayFPT *with polynomial space, and*
- ENUM-WEAK-$\mathcal{C}$-BACKDOORS$^{\text{SIZE}} \in$ DelayFPT *with exponential space.*

Proof. The set of operations Ω then contains functions that replace a specific variable $i \in \mathbb{N}$ by a truth value $t \in \{0,1\}$. A solution then encodes the chosen backdoor sets together with the required assignment. Then, Proposition 3 yields ENUM-MIN-WEAK-$\mathcal{C}$BACKDOORS$^{\text{LEX}}$, resp., ENUM-MIN-WEAK-$\mathcal{C}$-BACKDOORS$^{\text{SIZE}}$ being FPT-enumerable. As the consistency check for solutions is in polynomial time, applying Corollary 7 completes the proof. □

In the following result, we will examine the parametrised enumeration complexity of the task to enumerate all strong $\mathcal{C}$-backdoor sets of a given 3CNF formula for some clause-defined base class $\mathcal{C}$. Crucially, every strong backdoor set has to contain at least one variable from a clause that is not in C which relates to 'hitting all bad clauses' like in the definition of deletion backdoors (see Def. 15).

Theorem 5. *For every clause-defined base class $\mathcal{C}$ and input formulas in 3-CNF:*

- ENUM-STRONG-$\mathcal{C}$-BACKDOORS$^{\text{LEX}} \in$ DelayFPT *with polynomial space, and*
- ENUM-STRONG-$\mathcal{C}$-BACKDOORS$^{\text{SIZE}} \in$ DelayFPT *with exponential space.*

Proof. We show that for every clause-defined base class $\mathcal{C}$ and input formulas in 3-CNF, the problem MIN-STRONG-$\mathcal{C}$-BACKDOORS is FPT-enumerable. Indeed, we only need to branch on the variables from a clause $C \notin \mathcal{C}$ and remove the corresponding literals over the considered variable from ϕ. The size of the branching tree is at most 3^k. As for base classes the satisfiability test is in P, this yields an FPT-algorithm. The neighbourhood function $\mathcal{N}(x, S)$ for $x = (\phi, k)$ is defined to be the set of the pairwise unions of all minimal strong $\mathcal{C}$-backdoors of $(\phi[(S \cup \{x_i\})], k - |S| - 1)$ together with $S \cup \{x_i\}$ for all variables $x_i \notin S$. If $\text{Vars}(\phi) = \{x_1, \ldots, x_n\}$, then the operations are $\omega_i \colon \phi \mapsto \phi(0/x_i) \wedge \phi(1/x_i)$. As applying the functions ω_i happens only with respect to the backdoor size k, which is the parameter, the formula size increases by an exponential factor in the parameter only. This yields the preconditions for Corollary 7 constituting the proof. □

4.3. Weighted Satisfiability Problems

Finally, we consider satisfiability questions for formulas in the Schaefer framework [44]. A *constraint language* Γ is a finite set of relations. A Γ*-formula* ϕ, is a conjunction of constraints using only relations from Γ and, consequently, is a quantifier-free first order formula.

As opposed to the approach of Creignou et al. [12], who examined maximum satisfiability, we now focus on the problem MINONES-SAT(Γ) defined below.

Definition 18 (Minimality). *Given a propositional formula ϕ and an assignment θ over the variables in ϕ with $\theta \models \phi$, we say θ is* minimal *if there does not exist an assignment $\theta' \subset \theta$ and $\theta' \models \phi$. The size $|\theta|$ of θ is the number of variables it sets to true.*

Formally, the problem of interest is defined with respect to any fixed constraint language Γ:

Problem:	MIN-MINONES-SAT$^{\text{SIZE}}(\Gamma)$
Input:	(ϕ, k), a Γ-formula ϕ, $k \in \mathbb{N}$.
Parameter:	The integer k.
Output:	Generate all inclusion-minimal satisfying assignments θ of ϕ with $\lvert\theta\rvert \leq k$ by non-decreasing size.

Similarly, the problem ENUM-MINONES-SAT(Γ) asks for all satisfying assignments θ of ϕ with $|\theta| \leq k$. In this context, the operations in Ω are functions that replace the variable with index $i \in \mathbb{N}$ by true.

Theorem 6. *For all constraint languages* Γ*, we have:* MIN-MINONES-SAT$^{\text{SIZE}}(\Gamma)$ *is* FPT*-enumerable and* ENUM-MINONES-SAT$^{\text{SIZE}}(\Gamma) \in$ DelayFPT *with exponential space.*

Proof. For the first claim we can simply compute the minimal assignments by a straight forward branching algorithm: initially, begin with the all 0-assignment, then consider all unsatisfied clauses in turn and flip one of the occurring variables to true. The second claim follows by a direct application of Corollary 7. □

5. Conclusion

We presented FPT-delay ordered enumeration algorithms for a large variety of problems, such as cluster editing, chordal completion, closest-string, and weak and strong backdoors. An important point of our paper is that we propose a general strategy for efficient enumeration. This is rather rare in the literature, where algorithms are usually devised individually for specific problems. In particular, our scheme yields DelayFPT algorithms for all graph modification problems that are characterised by a finite set of forbidden patterns.

Initially, we focussed on graph-theoretic problems. Afterwards, the generic approach we presented, covers problems which are not only of a graph-theoretic nature. Here, we defined general modification problems, detached from graphs and constructed generic enumeration algorithms for arising problems in the world of strings, numbers, formulas, constraints, etc.

As an observation we would like to mention that the DelayFPT algorithms presented in this paper require exponential space due to the inherent use of the priority queues to achieve *ordered* enumeration. An interesting question, continuing research of Meier [45], is whether there is a method which requires less space but uses a comparable delay between the output of solutions and still obeys the underlying order on solutions.

Author Contributions: Conceptualization, Nadia Creignou, Raïda Ktari, Arne Meier, Julian-Steffen Müller, Frédéric Olive and Heribert Vollmer; Funding acquisition, Nadia Creignou, Arne Meier and Heribert Vollmer; Methodology, Nadia Creignou, Raïda Ktari, Arne Meier, Julian-Steffen Müller, Frédéric Olive and Heribert Vollmer; Supervision, Nadia Creignou, Arne Meier and Heribert Vollmer; Writing – original draft, Nadia Creignou, Raïda Ktari, Arne Meier, Frédéric Olive and Heribert Vollmer; Writing – review & editing, Nadia Creignou, Raïda Ktari, Arne Meier, Frédéric Olive and Heribert Vollmer.

Funding: This research was funded by Deutsche Forschungsgemeinschaft (ME 4279/1-2) and the French Agence Nationale de la Recherche (AGGREG project reference ANR-14-CE25-0017).

Conflicts of Interest: The authors declare no conflict of interest.

1. Durand, A.; Schweikardt, N.; Segoufin, L. Enumerating answers to first-order queries over databases of low degree. Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014; Hull, R.; Grohe, M., Eds. ACM, 2014, pp. 121–131. doi:10.1145/2594538.2594539.
2. Fogaras, D.; Rácz, B. A Scalable Randomized Method to Compute Link-Based Similarity Rank on the Web Graph. Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops

PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers; Lindner, W.; Mesiti, M.; Türker, C.; Tzitzikas, Y.; Vakali, A., Eds. Springer, 2004, Vol. 3268, *Lecture Notes in Computer Science*, pp. 557–567. doi:10.1007/978-3-540-30192-9_55.

3. Acuña, V.; Milreu, P.V.; Cottret, L.; Marchetti-Spaccamela, A.; Stougie, L.; Sagot, M. Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks. *Bioinformatics* **2012**, *28*, 2474–2483. doi:10.1093/bioinformatics/bts423.
4. Dill, K.A.; Lucas, A.; Hockenmaier, J.; Huang, L.; Chiang, D.; Joshi, A.K. Computational linguistics: A new tool for exploring biopolymer structures and statistical mechanics. *Polymer* **2007**, *48*, 4289 – 4300. doi:https://doi.org/10.1016/j.polymer.2007.05.018.
5. Johnson, D.S.; Papadimitriou, C.H.; Yannakakis, M. On Generating All Maximal Independent Sets. *Information Processing Letters* **1988**, *27*, 119–123.
6. Sörensen, K.; Janssens, G.K. An algorithm to generate all spanning trees of a graph in order of increasing cost. *Pesquisa Operacional* **2005**, *25*, 219–229.
7. Creignou, N.; Hébrard, J.J. On generating all solutions of generalized satisfiability problems. *Theoretical Informatics and Applications* **1997**, *31*, 499–511.
8. Creignou, N.; Olive, F.; Schmidt, J. Enumerating All Solutions of a Boolean CSP by Non-decreasing Weight. Proc. SAT. Springer, 2011, Vol. 6695, *LNCS*, pp. 120–133.
9. Fernau, H. On parameterized enumeration. *Computing and Combinatorics* **2002**.
10. Damaschke, P. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science* **2006**, *351*, 337–350.
11. Fomin, F.V.; Saurabh, S.; Villanger, Y. A Polynomial Kernel for Proper Interval Vertex Deletion. *SIAM Journal Discrete Mathematics* **2013**, *27*, 1964–1976.
12. Creignou, N.; Meier, A.; Müller, J.; Schmidt, J.; Vollmer, H. Paradigms for Parameterized Enumeration. *Theory of Computing Systems* **2017**, *60*, 737–758. doi:10.1007/s00224-016-9702-4.
13. Shamir, R.; Sharan, R.; Tsur, D. Cluster graph modification problems. *Discrete Applied Mathematics* **2004**, *114*, 173–182.
14. Yannakakis, M. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods* **1981**, *2*, 77–79. doi:10.1137/0602010.
15. Yannakakis, M. Node- and edge-deletion NP-complete problems. Proc. STOC, 1978, pp. 253–264.
16. Gramm, J.; Niedermeier, R.; Rossmanith, P. Fixed-Parameter Algorithms for CLOSEST STRING and Related Problems. *Algorithmica* **2003**, *37*, 25–42.
17. Williams, R.; Gomes, C.P.; Selman, B. Backdoors To Typical Case Complexity. IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003, 2003, pp. 1173–1178.
18. Kröll, M.; Pichler, R.; Skritek, S. On the Complexity of Enumerating the Answers to Well-designed Pattern Trees. 19th International Conference on Database Theory (ICDT 2016); Martens, W.; Zeume, T., Eds.; Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2016; Vol. 48, *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 22:1–22:18. doi:10.4230/LIPIcs.ICDT.2016.22.
19. Bentert, M.; Fluschnik, T.; Nichterlein, A.; Niedermeier, R. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences* **2019**, *103*, 61–77. doi:10.1016/j.jcss.2019.02.004.
20. Bökler, F.; Ehrgott, M.; Morris, C.; Mutzel, P. Output-sensitive complexity of multiobjective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis* **2017**, *24*, 25–36. doi:10.1002/mcda.1603.
21. Kröll, M.; Pichler, R.; Woltran, S. On the Complexity of Enumerating the Extensions of Abstract Argumentation Frameworks. Proceedings of the 26th International Joint Conference on Artificial Intelligence. AAAI Press, 2017, IJCAI'17, pp. 1145–1152.
22. Carbonnel, C.; Hebrard, E. On the Kernelization of Global Constraints. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017; Sierra, C., Ed. ijcai.org, 2017, pp. 578–584. doi:10.24963/ijcai.2017/81.
23. Schmidt, J. Enumeration: Algorithms and Complexity. Master's thesis, Leibniz Universität Hannover, 2009.
24. Hirai, Y.; Yamamoto, K. Balancing weight-balanced trees. *Journal of Functional Programming* **2011**, *21*, 287–307. doi:10.1017/S0956796811000104.

25. Bodlaender, H.L.; Heggernes, P.; Lokshtanov, D. Graph Modification Problems (Dagstuhl Seminar 14071). *Dagstuhl Reports* **2014**, *4*, 38–59. doi:10.4230/DagRep.4.2.38.
26. Garey, M.R.; Johnson, D.S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: New York, NY, USA, 1990.
27. Cai, L. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters* **1996**, *58*, 171–176.
28. Kaplan, H.; Shamir, R.; Tarjan, R.E. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing* **1999**, *28*, 1906–1922.
29. Brandtstädt, A.; Le, V.B.; Spinrad, J.P. *Graph Classes: A Survey*; Monographs on Discrete Applied Mathematics, SIAM: Philadelphia, 1988.
30. Avis, D.; Fukuda, K. Reverse search for enumeration. *Discrete Applied Mathematics* **1996**, *65*, 21–46.
31. Frances, M.; Litman, A. On covering problems of codes. *Theory of Computing Systems* **1997**, *30*, 113–119.
32. Pfandler, A.; Rümmele, S.; Szeider, S. Backdoors to Abduction. Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13); Rossi, F., Ed.; , 2013; pp. 1046–1052.
33. Fichte, J.K.; Szeider, S. Backdoors to tractable answer set programming. *Artificial Intelligence* **2015**, *220*, 64–103. doi:10.1016/j.artint.2014.12.001.
34. Fichte, J.K.; Szeider, S. Backdoors to Normality for Disjunctive Logic Programs. *ACM Trans. Comput. Log.* **2015**, *17*, 7:1–7:23. doi:10.1145/2818646.
35. Dvořák, W.; Ordyniak, S.; Szeider, S. Augmenting tractable fragments of abstract argumentation **2012**. *186*, 157–173. doi:10.1016/j.artint.2012.03.002.
36. Fichte, J.K.; Meier, A.; Schindler, I. Strong Backdoors for Default Logic. Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings, 2016, pp. 45–59. doi:10.1007/978-3-319-40970-2_4.
37. Meier, A.; Ordyniak, S.; Ramanujan, M.S.; Schindler, I. Backdoors for Linear Temporal Logic. *Algorithmica* **2019**, *81*, 476–496. doi:10.1007/s00453-018-0515-5.
38. Kronegger, M.; Ordyniak, S.; Pfandler, A. Backdoors to planning. *Artificial Intelligence* **2019**, *269*, 49–75. doi:10.1016/j.artint.2018.10.002.
39. Ganian, R.; Ramanujan, M.S.; Szeider, S. Combining Treewidth and Backdoors for CSP. 34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany; Vollmer, H.; Vallée, B., Eds. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, Vol. 66, *LIPIcs*, pp. 36:1–36:17. doi:10.4230/LIPIcs.STACS.2017.36.
40. Gaspers, S.; Misra, N.; Ordyniak, S.; Szeider, S.; Zivny, S. Backdoors into heterogeneous classes of SAT and CSP. *Journal of Computer and System Sciences* **2017**, *85*, 38–56. doi:10.1016/j.jcss.2016.10.007.
41. Nishimura, N.; Ragde, P.; Szeider, S. Solving #SAT using vertex covers. *Acta Informatica* **2007**, *44*, 509–523. doi:10.1007/s00236-007-0056-x.
42. Szeider, S. Matched Formulas and Backdoor Sets. *JSAT* **2009**, *6*, 1–12.
43. Gaspers, S.; Szeider, S. Backdoors to Satisfaction. In *The Multivariate Algorithmic Revolution and Beyond*; Springer Berlin Heidelberg, 2012; Vol. 7370, *LNCS*, pp. 287–317. doi:10.1007/978-3-642-30891-8_15.
44. Schaefer, T.J. The Complexity of Satisfiability Problems. Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA; Lipton, R.J.; Burkhard, W.A.; Savitch, W.J.; Friedman, E.P.; Aho, A.V., Eds. ACM, 1978, pp. 216–226. doi:10.1145/800133.804350.
45. Meier, A. Enumeration in Incremental FPT-Time. *CoRR* **2018**, *abs/1804.07799*, [1804.07799].