

# Enumerating minimal dominating sets in $K_t$ -free graphs and variants\*

Marthe Bonamy<sup>1</sup>, Oscar Defrain<sup>2</sup>, Marc Heinrich<sup>3</sup>,  
Michał Pilipczuk<sup>4</sup>, and Jean-Florent Raymond<sup>5</sup>

<sup>1</sup>*CNRS, LaBRI, Université de Bordeaux, France.*

<sup>2</sup>*LIMOS, Université Clermont Auvergne, France.*

<sup>3</sup>*LIRIS, Université Claude-Bernard, Lyon, France.*

<sup>4</sup>*University of Warsaw, Poland.*

<sup>5</sup>*CNRS, LIMOS, Université Clermont Auvergne, France.*

March 6, 2020

## Abstract

It is a long-standing open problem whether the minimal dominating sets of a graph can be enumerated in output-polynomial time. In this paper we investigate this problem in graph classes defined by forbidding an induced subgraph. In particular, we provide output-polynomial time algorithms for  $K_t$ -free graphs and for several related graph classes. This answers a question of Kanté et al. about enumeration in bipartite graphs.

## 1 Introduction

Countless algorithmic problems in graph theory require to detect a structure with prescribed properties in an input graph. Rather than finding one such object, it is sometimes more desirable to generate all of them. This is for instance useful in certain applications to database search [YYH05], network analysis [GK07], bioinformatics [Dam06, Mar15], and cheminformatics [Bar93]. Enumeration algorithms for graph problems seem to have been first mentioned in the early 70's with the pioneer works of Tiernan [Tie70] and Tarjan [Tar73] on cycles in directed graphs, and of Akkoyunlu [Akk73] on maximal cliques in undirected graphs. However, they already appeared in disguise in earlier works [PU59, Mar64]. To this date, several intriguing questions on the topic remain unsolved. We refer the reader to [Mar15, Chapter 2] and [Str19] for more in-depth introductions to enumeration algorithms, and to [Was16] for a listing of enumeration algorithms and problems.

---

\*A preliminary version of this article appeared in the proceedings of the 36<sup>th</sup> Symposium on Theoretical Aspects of Computer Science (STACS 2019) [BDHR19]. The first author has been supported by the ANR project GrR ANR-18-CE40-0032. The second author has been supported by the ANR project GraphEn ANR-15-CE40-0009. The fourth author is supported by project TOTAL, which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 677651. The last author has been supported by the ERC consolidator grant DISTRUCT-648527.

The objects we wish to enumerate in this paper are the (inclusion-wise) minimal dominating sets of a given graph. In general, the number of these objects may grow exponentially with the order  $n$  of the input graph. Therefore, in stark contrast to decision or optimization problems, looking for a running time polynomially bounded by  $n$  is not a reasonable, let alone meaningful, efficiency criterion. Rather, we aim here for so-called *output-polynomial* time algorithms [JYP88], whose running time is polynomially bounded by the size of both the input and output data, and refer to [FGPS08, CHvHK13, GKLS19] for input exponential-time algorithms for the problem we consider in this paper.

Since dominating sets are among the most studied objects in graph theory and algorithms, their enumeration (and counting) have attracted an increasing attention over the past 10 years. The problem of enumerating minimal dominating sets (hereafter referred to as DOM-ENUM) has a notable feature: it is equivalent to the extensively studied problem TRANS-ENUM. In TRANS-ENUM, one is given a hypergraph  $\mathcal{H}$  (i.e., a collection of subsets, called *hyperedges*, of elements called *vertices*) and is asked to enumerate all the (inclusion-wise) minimal *transversals* of  $\mathcal{H}$  (i.e., the inclusion-wise minimal sets of vertices that meet every hyperedge). It is not hard to see that DOM-ENUM is a particular case of TRANS-ENUM: the minimal dominating sets of a graph  $G$  are exactly the minimal transversals of the hypergraph of closed neighborhoods of  $G$ . Conversely, Kanté, Limouzy, Mary, and Nourine proved that every instance of TRANS-ENUM can be reduced to a co-bipartite<sup>1</sup> instance of DOM-ENUM [KLMN14]. Currently, the best output-sensitive algorithm for TRANS-ENUM is due to Fredman and Khachiyan and runs in output quasi-polynomial time [FK96]. It is a long-standing open problem whether this complexity bound can be improved (see for instance the surveys [EG02, EMG08]). Therefore, the equivalence between the two problems is an additional motivation to study DOM-ENUM, with the hope that techniques from graph theory could be used to obtain new results on TRANS-ENUM.

So far, output-polynomial time algorithms have been obtained for DOM-ENUM in several classes of graphs, including planar graphs and degenerate graphs [EGM03], classes of graphs of bounded treewidth, cliquewidth [Cou09], or LMIM-width [GHK<sup>+</sup>18], path graphs and line graphs [KLMN12], interval graphs and permutation graphs [KLM<sup>+</sup>13], split graphs [KLM<sup>+</sup>15], graphs of girth at least 7 [GHKV15], chordal graphs [KLM<sup>+</sup>15], and chordal bipartite graphs [GHK<sup>+</sup>16]. A succinct survey of results on DOM-ENUM can be found in [KN16] and [GHK<sup>+</sup>16].

In this paper, we investigate the complexity of DOM-ENUM in graph classes defined by forbidding an induced subgraph  $H$ , hereafter referred to as  $H$ -free graphs. For every  $t \in \mathbb{N}$ , we denote by  $K_t$  the complete graph on  $t$  vertices, by  $K_t - e$  the graph obtained by removing any edge in  $K_t$  and by  $K_t + K_2$  the disjoint union of  $K_t$  and  $K_2$ . Our main result is the following.

**Theorem 1.1.** *There is an algorithm enumerating, for every fixed  $t \in \mathbb{N}$ , the minimal dominating sets in  $(K_t + K_2)$ -free graphs in output-polynomial time and polynomial space.*

In particular, this yields an output-polynomial time algorithm for  $K_t$ -free graphs. A notable special case is that of bipartite graphs, where the question of the existence of an output-polynomial time algorithm for DOM-ENUM was explicitly stated in [KN16] and later papers [KLM<sup>+</sup>15, GHK<sup>+</sup>16]. We stress that we provide in the proof of Theorem 1.1 a single algorithm that deals with all values of  $t$  and that this algorithm does not require the knowledge of  $t$ . We discuss the complexity in greater details in Sections 4 and 5.

---

<sup>1</sup>The complement of a bipartite graph.

In order to push our techniques to their limits, we investigate cases that are close to but not covered by Theorem 1.1. Namely, we consider two particular choices of the graph  $H$ : the *paw*, which is the graph obtained by adding a vertex of degree one to  $K_3$  (i.e.,  $H = \text{paw}$ ), and  $K_4 - e$ , also known as *diamond* graph (i.e.,  $H = \text{diamond}$ ). We combine our main tools with some ad hoc techniques to handle those two cases, thus obtaining the following.

**Theorem 1.2.** *There is an algorithm enumerating minimal dominating sets in paw-free (resp. diamond-free) graphs in output-polynomial time and polynomial space.*

Our algorithms first decompose the input graph by successively removing closed neighborhoods in the fashion of [EGM03]. We then follow this decomposition to construct partial minimal dominating sets, adding the neighborhoods back one after the other. A crucial point of this approach, known as *ordered generation*, is that we can relate the enumeration of potential extensions of a partial minimal dominating set to the DOM-ENUM problem in a simpler class.

The paper is organized as follows. In Section 2 we give the necessary definitions. The graph decompositions that we use, called *peelings*, are introduced in Section 3 along with their main properties. In Section 4, we give an algorithm for DOM-ENUM that runs in output-polynomial time in triangle-free graphs with better time bound than that coming from Theorem 1.1. A generalization of this algorithm for  $K_t$ -free graphs is given in Section 5 (Theorem 5.4). This algorithm is then extended to  $(K_t + K_2)$ -free graphs in Section 6 (Theorem 6.1). In the same section, algorithms are given for diamond-free graphs (Theorem 6.7) and paw-free graphs (Theorem 6.11), i.e., the two cases of Theorem 1.2. We discuss in Section 7 the obstacles to stronger theorems using the same tools. Finally, we conclude with possible future research directions in Section 8.

## 2 Preliminaries

**Graphs.** All graphs in this paper are finite, undirected, simple, and loopless. If  $G = (V(G), E(G))$  is a graph, then  $V(G)$  is its set of vertices and  $E(G) \subseteq \{\{x, y\} \mid x, y \in V(G), x \neq y\}$  is its set of edges. Edges are denoted by  $xy$  (or  $yx$ ) instead of  $\{x, y\}$ . We assume that vertices are assigned distinct indices; these will be used to choose vertices in a deterministic way, typically selecting the vertex of smallest index. A *clique* (respectively an *independent set*) in a graph  $G$  is a set of pairwise adjacent (respectively non-adjacent) vertices. We note  $\omega(G)$  the size of a largest clique in  $G$ . The subgraph of  $G$  induced by  $X \subseteq V(G)$ , denoted by  $G[X]$ , is the graph  $(X, E(G) \cap \{\{x, y\} \mid x, y \in X, x \neq y\})$ ;  $G - X$  is the graph  $G[V(G) \setminus X]$ . For every graph  $H$ , we say that  $G$  is  $H$ -free if no induced subgraph of  $G$  is isomorphic to  $H$ . If a vertex  $v \in V(G)$  is adjacent to every vertex of a set  $S \subseteq V(G)$ , we say that  $v$  is *complete* to  $S$ .

If the vertex set of a graph  $G$  can be partitioned into one part inducing a clique and one part inducing an independent set (respectively two independent sets, two cliques), we say that  $G$  is a *split* (respectively *bipartite*, *co-bipartite*) graph. If  $f$  is a function, we write  $f(n) = \text{poly}(n)$  when there is a constant  $c \in \mathbb{N}$  such that  $f(n) \in O(n^c)$ .

**Neighbors and domination.** Let  $G$  be a graph and  $x \in V(G)$ . We write  $N(x)$  for the set of *neighbors* of  $x$  in  $G$  defined by  $N(x) = \{y \in V(G) \mid xy \in E(G)\}$ ;  $N[x]$  is the *closed neighborhood* of  $x$  defined by  $N[x] = N(x) \cup \{x\}$ . For a given  $X \subseteq V(G)$ , we note

$N[X] = \bigcup_{x \in X} N[x]$  and  $N(X) = N[X] \setminus X$ . Let  $D$  be a set of vertices of  $G$ . We say that  $D$  *dominates* a subset  $S \subseteq V(G)$  if  $S \subseteq N[D]$ . It *minimally dominates*  $S$  if no proper subset of  $D$  dominates  $S$ . The set  $D$  is a (*minimal*) *dominating set* of  $G$  if it (minimally) dominates  $V(G)$ . The set of all minimal dominating sets of  $G$  is denoted by  $\mathcal{D}(G)$  and the problem of enumerating  $\mathcal{D}(G)$  given  $G$  is denoted by DOM-ENUM.

Let  $S \subseteq V(G)$ . A vertex  $y \in V(G)$  is said to be a *private neighbor* of some  $x \in S$  if it is only dominated by  $x$  in  $S$ , i.e., if  $y \in N[S]$  but  $y \notin N[S \setminus \{x\}]$ . Note that  $x$  can be its own private neighbor. The set of private neighbors of  $x \in S$  in  $G$  is denoted by  $\text{Priv}_G(S, x)$  and we drop the subscript when it can be inferred from the context. Observe that  $S$  is a minimal dominating set of  $G$  if and only if  $V(G) \subseteq N[S]$  and for every  $x \in S$ ,  $\text{Priv}(S, x) \neq \emptyset$ .

**Enumeration.** In this paper, to measure time and space complexity we assume the RAM model, where any integer can be stored in a single register and arithmetic operations on integers have unit cost [Str19]. The aim of graph enumeration algorithms is to generate a set  $\mathcal{X}(G)$  of objects related to a graph  $G$ . We say that an algorithm enumerating  $\mathcal{X}(G)$  on input being an  $n$ -vertex graph  $G$  is running in *output-polynomial* time if its running time is polynomially bounded by the sizes of the input and output data, i.e.,  $n + |\mathcal{X}(G)|$ . If an algorithm enumerates  $\mathcal{X}(G)$  by spending  $\text{poly}(n)$ -time (respectively  $O(n)$ -time) before it outputs the first element, between two output elements, and after it outputs the last element, then we say that it runs with *polynomial delay* (respectively *linear delay*). It is easy to see that every polynomial delay algorithm is also output-polynomial. Note, however, that there exist problems that admit output-polynomial time algorithms but no polynomial delay ones, unless  $\text{TFNP} = \text{FP}$  [Str19]. When discussing the space used by an enumeration algorithm, we mean the working space and we ignore the space where the solutions are output. If the existence of an output-polynomial time algorithm for a problem implies the existence of one for DOM-ENUM, we say that this problem is DOM-ENUM-hard. As mentioned in the introduction, we have the following.

**Theorem 2.1** (Kanté et al. [KLMN14]). *DOM-ENUM restricted to co-bipartite graphs is DOM-ENUM-hard.*

### 3 Ordered generation in bicolored graphs

In this section, we give a general procedure that will be used in the rest of this paper for the enumeration of minimal dominating sets in  $K_t$ -free graphs and in related graph classes. This procedure will construct minimal dominating sets one neighborhood at a time, in a variant of what is known as the *backtrack search technique* in [RT75, FLM97, SM19], and referred to as *ordered generation* in [EGM03].

In what follows, we find it more convenient to deal with the slightly more general setting of domination in bicolored graphs. A *bicolored graph* is a graph together with a subset of its vertex set. For a graph  $G$  and a subset  $A \subseteq V(G)$ , we denote by  $G(A)$  the bicolored graph  $G$  with prescribed set  $A$ . We also say that  $G$  has *bicoloring*  $(A, V(G) \setminus A)$ . Then, a *dominating set of  $G(A)$*  is a set  $D \subseteq V(G)$  that dominates  $A$ , i.e., such that  $A \subseteq N[D]$ . It is (inclusion-wise) *minimal* if it does not contain any dominating set of  $G(A)$  as a proper subset. Intuitively, the vertices of  $G - A$  may be used in the dominating set, but do not need to be dominated. For every graph  $G$  and subset  $A \subseteq V(G)$ , we

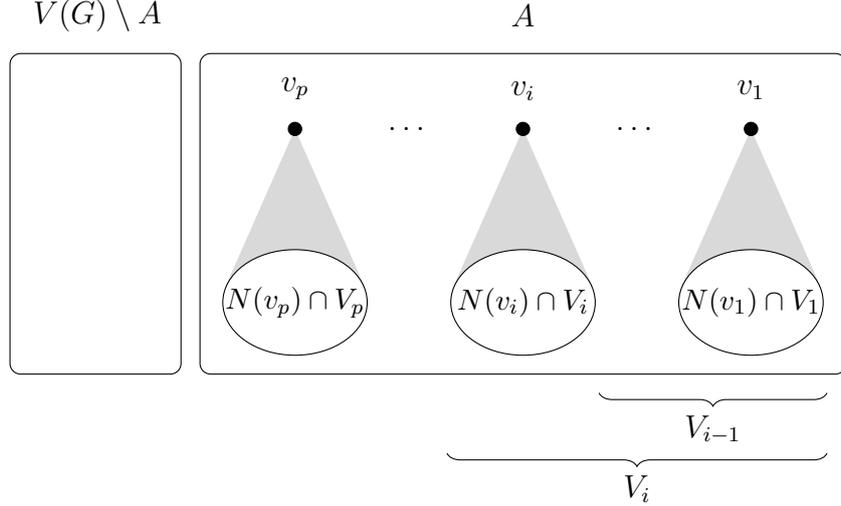


Figure 1: Representation of a peeling of a bicolored graph  $G(A)$  constructed by iteratively removing  $v_i$ 's and their neighborhoods, for  $i$  from  $p$  to 1. Note that vertices that are effectively removed at step  $i$  are those of  $N[v_i] \cap V_i$ , as vertices in  $N[v_i] \setminus V_i$  have already been removed at a previous step. A crucial property is that  $v_i$  has no neighbor in  $V_{i-1}$ .

denote by  $\mathcal{D}(G, A)$  the set of minimal dominating sets of  $G(A)$ . Then  $\mathcal{D}(G, A) = \mathcal{D}(G)$  whenever  $A = V(G)$ .

A *peeling* of a bicolored graph  $G(A)$  is a sequence of vertex sets  $(V_0, \dots, V_p)$  such that  $V_p = A$ ,  $V_0 = \emptyset$ , and for every  $i \in \{1, \dots, p\}$ , there is a vertex  $v_i \in V_i$  such that

$$V_{i-1} = V_i \setminus N[v_i].$$

We call  $(v_1, \dots, v_p)$  the *vertex sequence* of the peeling. It is straightforward to see that given a bicolored graph  $G(A)$ , any peeling of  $G(A)$  can be computed in  $O(n^2)$  time and space: start with the whole set  $A$ , and as long as  $A$  remains non-empty, pick a vertex  $v$  in it and remove  $N[v]$  from  $A$ . The representation of a peeling is given in Figure 1.

In the remaining of this section, we consider a bicolored graph  $G(A)$ , together with a fixed peeling  $(V_0, \dots, V_p)$  of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ . Observe that  $\mathcal{D}(G, V_p) = \mathcal{D}(G, A)$ . We now define the relation that will be used by our algorithm to enumerate the minimal dominating sets of  $G(A)$  without repetition. Recall that the sets of  $\mathcal{D}(G, V_i)$  may contain vertices of  $G - V_i$ , which is a crucial point.

**Definition 3.1.** Let  $i \in \{0, \dots, p-1\}$  and  $D \in \mathcal{D}(G, V_{i+1})$ . We define  $\text{Parent}(D, i+1)$  as the pair  $(D^*, i)$  where  $D^*$  is obtained from  $D$  by exhaustively applying the following operation: as long as there exists a vertex  $x$  in  $D$  satisfying  $\text{Priv}(D, x) \cap V_i = \emptyset$ , remove from  $D$  the vertex of smallest index with this property.

Clearly, there is a unique way to build  $\text{Parent}(D, i+1)$  given  $D$  and  $i$ . By construction, the obtained set  $D^*$  is a minimal dominating set of  $G(V_i)$ . Hence, every set in  $\mathcal{D}(G, V_{i+1})$  can be obtained by completing some  $D^*$  in  $\mathcal{D}(G, V_i)$ ; we develop this point below.

**Proposition 3.2.** Let  $i \in \{0, \dots, p-1\}$  and  $D^* \in \mathcal{D}(G, V_i)$ . Then:

- (i) if  $D^*$  dominates  $V_{i+1}$  then  $D^* \in \mathcal{D}(G, V_{i+1})$  and  $\text{Parent}(D^*, i+1) = (D^*, i)$ ;
- (ii) otherwise,  $D^* \cup \{v_{i+1}\} \in \mathcal{D}(G, V_{i+1})$  and  $\text{Parent}(D^* \cup \{v_{i+1}\}, i+1) = (D^*, i)$ .

*Proof.* First note that since  $D^* \in \mathcal{D}(G, V_i)$ , for all  $x \in D^*$  we have  $\text{Priv}(D^*, x) \cap V_i \neq \emptyset$ , implying also  $\text{Priv}(D^*, x) \cap V_{i+1} \neq \emptyset$ . Hence, if  $D^*$  dominates  $V_{i+1}$  then in fact  $D^*$  is a minimal dominating set of  $G(V_{i+1})$  and thus  $D^* \in \mathcal{D}(G, V_{i+1})$ . Since  $\text{Priv}(D^*, x) \cap V_i \neq \emptyset$  for all  $x \in D^*$ , we then have that  $\text{Parent}(D^*, i+1) = (D^*, i)$  directly from the definition.

Suppose now that  $D^*$  does not dominate  $V_{i+1}$ , and observe that then  $D = D^* \cup \{v_{i+1}\}$  does. Moreover,  $\text{Priv}(D, v_{i+1}) \cap V_{i+1} \neq \emptyset$ . Since  $v_{i+1}$ , by the definition of the peeling, is not adjacent to any vertex in  $V_i$ , it cannot steal any private neighbor from the elements of  $D^*$ , i.e.,  $\text{Priv}(D^*, x) \cap V_i \neq \emptyset$  implies  $\text{Priv}(D^* \cup \{v_{i+1}\}, x) \cap V_i \neq \emptyset$  for any  $x \in D^*$ . Hence  $\text{Priv}(D, x) \cap V_{i+1} \neq \emptyset$  for all  $x \in D$ . Now, note that since  $v_{i+1}$  does not steal private neighbors from the elements of  $D^*$ , it is indeed the only node in  $D$  with no private neighbors in  $V_i$ , and it is removed when constructing  $\text{Parent}(D, i+1)$ . Hence  $\text{Parent}(D, i+1) = (D^*, i)$ , as claimed.  $\square$

The  $\text{Parent}$  function as introduced in Definition 3.1 defines a tree on vertex set

$$\{(D, i) \mid i \in \{0, \dots, p\}, D \in \mathcal{D}(G, V_i)\},$$

with leaves  $\{(D, p) \mid D \in \mathcal{D}(G, A)\}$  and root  $(\emptyset, 0)$  (the empty set being the only minimal dominating set of the empty vertex set  $V_0$ ). Our algorithms will search this tree in order to enumerate the minimal dominating sets of  $G(A)$ . Proposition 3.2 guarantees that for every  $i < p$  and every  $D^* \in \mathcal{D}(G, V_i)$ , the pair  $(D^*, i)$  is the parent of some  $(D, i+1)$  with  $D \in \mathcal{D}(G, V_{i+1})$  (possibly  $D = D^*$ ). Consequently, every branch of the tree leads to a different minimal dominating set of  $G(A)$ . In particular, for every  $i \in \{0, \dots, p-1\}$  we have

$$|\mathcal{D}(G, V_i)| \leq |\mathcal{D}(G, V_{i+1})| \leq |\mathcal{D}(G, A)|. \quad (1)$$

Given a set  $D^* \in \mathcal{D}(G, V_i)$ , we now focus on the enumeration of all  $D \in \mathcal{D}(G, V_{i+1})$  such that  $\text{Parent}(D, i+1) = (D^*, i)$ . Any (inclusion-wise) minimal set  $X \subseteq V(G)$  such that  $V_{i+1} \subseteq N[D^* \cup X]$  will be called a *candidate extension* of  $(D^*, i)$ . In other words,  $X$  is a candidate extension of  $(D^*, i)$  if and only if it is a minimal dominating set of the bicolored graph  $G$  with prescribed set  $V_{i+1} \setminus N[D^*]$ . Then, we denote by  $\mathcal{C}(D^*, i)$  the set of all candidate extensions of  $(D^*, i)$ , i.e.,

$$\mathcal{C}(D^*, i) \stackrel{\text{def}}{=} \mathcal{D}(G, V_{i+1} \setminus N[D^*]). \quad (2)$$

Observe that if  $(D, i+1)$  has  $(D^*, i)$  as its parent, then  $D \setminus D^*$  is candidate extension of  $(D^*, i)$ . From Proposition 3.2, we also know that one of  $(D^*, i+1)$  and  $(D^* \cup \{v_{i+1}\}, i+1)$  has  $(D^*, i)$  as its parent, hence that either  $\emptyset$  or  $\{v_{i+1}\}$  is a candidate extension of  $(D^*, i)$ . Note that we have no guarantee that any other candidate extension forms a minimal dominating set of  $V_{i+1}$ , together with  $D^*$ . We show that it is still reasonable to test each of the candidate extensions even though  $D^*$  might have a unique child.

**Lemma 3.3.** *Let  $H(B)$  be a bicolored graph and  $D \subseteq V(H)$ . Then*

$$|\mathcal{D}(H, B \setminus N[D])| \leq |\mathcal{D}(H, B)|.$$

*Proof.* We argue that for every  $X \in \mathcal{D}(H, B \setminus N[D])$  we can find a set  $D_X \in \mathcal{D}(H, B)$  so that the sets  $D_X$  are pairwise different for different  $X$ ; this assertion immediately implies the desired inequality. For this, we define  $D_X$  as any minimal dominating set of  $H(B)$  that is a subset of  $D \cup X$ ; such a set exists as  $D \cup X$  dominates  $B$ . By definition, every vertex of  $X$  has a private neighbor in  $B \setminus N[D]$  so we have  $X \subseteq D_X$ . Moreover, since  $X$  is a minimal dominating set of  $B \setminus N[D]$ ,  $X$  is disjoint with  $D$ . We conclude that  $X = D_X \setminus D$ , and hence that the sets  $D_X$  are pairwise different for different  $X$ .  $\square$

As a consequence of Lemma 3.3 and Inequality (1), we have the following.

**Corollary 3.4.** *Let  $i \in \{0, \dots, p-1\}$  and  $D^* \in \mathcal{D}(G, V_i)$ . Then  $|\mathcal{C}(D^*, i)| \leq |\mathcal{D}(G, A)|$ .*

We conclude the ordered generation procedure with the following statement, which reduces the existence of an output-polynomial time algorithm enumerating  $\mathcal{D}(G, A)$  to the existence of one enumerating  $\mathcal{C}(D^*, i)$  for any  $i \in \{0, \dots, p-1\}$  and  $D^* \in \mathcal{D}(G, V_i)$ .

**Theorem 3.5.** *Let  $f: \mathbb{N}^2 \rightarrow \mathbb{N}$  and  $s: \mathbb{N} \rightarrow \mathbb{N}$  be two functions. Assume that there is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices, a peeling  $(V_0, \dots, V_p)$  of  $G(A)$ ,  $i \in \{0, \dots, p-1\}$ , and  $D^* \in \mathcal{D}(G, V_i)$ , enumerates  $\mathcal{C}(D^*, i)$  in time at most  $f(n, |\mathcal{D}(G, A)|)$  and space at most  $s(n)$ . Then there is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices, enumerates the set  $\mathcal{D}(G, A)$  in time*

$$O(n^4 d^2 + f(n, d) \cdot nd)$$

and space  $O(n \cdot s(n))$ , where  $d = |\mathcal{D}(G, A)|$ .

*Proof.* Let us assume that there exists an algorithm B that, given a bicolored graph  $G(A)$  on  $n$  vertices, a peeling  $(V_0, \dots, V_p)$  of  $G(A)$ ,  $i \in \{0, \dots, p-1\}$  and  $D^* \in \mathcal{D}(G, V_i)$ , enumerates  $\mathcal{C}(D^*, i)$  in time at most  $f(n, |\mathcal{D}(G, A)|)$  and space at most  $s(n)$ . Note that we may assume that  $s(n) \in \Omega(n)$ , as B needs to store its input. We describe an algorithm A that enumerates  $\mathcal{D}(G, A)$  within the specified time and space complexities.

The algorithm first checks whether  $A = \emptyset$  and, if so, returns  $\{\emptyset\}$ . Otherwise, it computes a peeling  $(V_0, \dots, V_p)$  of  $G(A)$  in time  $O(n^2)$  and using  $O(n^2)$  space. Recall that the Parent relation defines a tree  $T$  on vertex set

$$\{(D, i) \mid i \in \{0, \dots, p\}, D \in \mathcal{D}(G, V_i)\},$$

with leaves  $\{(D, p) \mid D \in \mathcal{D}(G, A)\}$  and root  $(\emptyset, 0)$ . Therefore, in order to enumerate  $\mathcal{D}(G, A)$ , it is enough for A to enumerate the leaves of  $T$ . To do so, the algorithm performs a depth-first search (DFS) of  $T$  outputting each visited leaf. For each node  $(D^*, i)$ ,  $i \in \{0, \dots, p-1\}$  of  $T$ , the algorithm runs B on input  $(G(A), (V_0, \dots, V_p), i, D^*)$  to generate  $\mathcal{C}(D^*, i)$  in time  $f(n, d)$  and space  $s(n)$ . For every  $X \in \mathcal{C}(D^*, i)$  generated by B, the algorithm tests whether  $D^* \cup X$  is a minimal dominating set of  $V_{i+1}$ , and whether  $\text{Parent}(D^* \cup X, i+1) = (D^*, i)$ . This requires  $O(n^3)$  steps per candidate extension, and a total working space of  $O(n)$ , disregarding the space needed to store the (globally fixed) graph  $G$ . As by Corollary 3.4 we have  $|\mathcal{C}(D^*, i)| \leq |\mathcal{D}(G, A)| = d$ , the total time spent by A at each node of  $T$  is bounded by  $O(n^3 d + f(n, d))$ . By Inequality (1) we have  $|V(T)| \leq pd$  and clearly  $p \leq n$ , so the total running time of A is bounded by

$$O(n^4 d^2 + f(n, d) \cdot nd).$$

Regarding the space, we observe that whenever we visit a node of  $T$ , we do not need to compute the whole set of its children. Instead, it is enough in order to continue the DFS to compute the next unvisited child only, which can be done using B and pausing it afterward. Therefore, when we visit some  $(D, i) \in V(T)$ , we only need to store the data of the  $i-1$  (paused) executions of B enumerating the children of the ancestors of  $(D, i)$ , plus the data of the algorithm enumerating the children of  $D$ , i.e.,  $i \cdot (O(n) + s(n))$  space. As  $s(n) \in \Omega(n)$ , the described algorithm uses  $O(n \cdot s(n))$  space, as claimed.  $\square$

## 4 Candidate extensions in triangle-free graphs

We show that candidate extensions can be enumerated in output-polynomial time in triangle-free graphs, which by Theorem 3.5 leads to an output-polynomial time algorithm enumerating minimal dominating sets in this class of graphs. In fact, our result holds in the more general context where only the graph induced by the set that needs to be dominated is required to be triangle-free, and not necessarily the whole graph, a point that is discussed in Section 7.

In the following, we consider a bicolored graph  $G(A)$  on  $n$  vertices. Moreover, we have a fixed peeling  $(V_0, \dots, V_p)$  of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ . Then, we consider

$$i \in \{0, \dots, p-1\}, \quad D^* \in \mathcal{D}(G, V_i),$$

and define  $\mathcal{C}(D^*, i)$  as in Equality (2) in Section 3. We will show how to enumerate  $\mathcal{C}(D^*, i)$  in output-polynomial time whenever  $G[A]$  is triangle-free.

Kanté, Limouzy, Mary, and Nourine gave the following characterization of minimal dominating sets in split graphs.

**Proposition 4.1** ([KLMN14]). *Let  $H$  be a split graph with vertices partitioned into an independent set  $S$  and a clique  $C$ , where  $S$  is taken to be (inclusion-wise) maximal. Then, for every  $D \in \mathcal{D}(H)$  the following holds:*

- (i)  $D \cap S = S \setminus N(D \cap C)$ , so in particular  $D$  is uniquely determined by its intersection with the clique; and
- (ii) for every  $x \in D$ ,  $\text{Priv}(D, x) \cap S \neq \emptyset$ .

Furthermore,  $\mathcal{D}(H)$  can be enumerated with delay  $O(n^2)$  and using  $O(n^2)$  space.

We can now use Proposition 4.1 to establish the following understanding of candidate extensions in terms of minimal dominating sets of an auxiliary split graph. The set  $S$  in the next lemma corresponds to the elements that, together with  $v_{i+1}$ , must be dominated by the candidate extensions of  $(D^*, i)$ . This situation is depicted in Figure 2.

**Lemma 4.2.** *Suppose that  $N(v_{i+1})$  is an independent set. Let  $S = V_{i+1} \setminus (N[D^*] \cup \{v_{i+1}\})$ ,  $C = N(S) \setminus \{v_{i+1}\}$ , and let  $H$  be the split graph obtained from  $G[S \cup C]$  by completing  $C$  into a clique. Then, every set  $X$  in  $\mathcal{C}(D^*, i)$  either belongs to  $\mathcal{D}(H)$ , or belongs to  $\mathcal{D}(H)$  after removing one vertex (i.e.,  $X \setminus \{u\} \in \mathcal{D}(H)$  for some  $u \in X$ ), or is such that  $X = \{v_{i+1}\}$ . Moreover,  $|\mathcal{D}(H)| \leq n \cdot |\mathcal{C}(D^*, i)| + 1$ .*

*Proof.* Consider any  $X \in \mathcal{C}(D^*, i)$ ,  $X \neq \{v_{i+1}\}$ . Then, by definition,  $X$  is a minimal dominating set of  $V_{i+1} \setminus N[D^*]$ . By assumption as  $V_{i+1} \setminus N[D^*] \subseteq N[v_{i+1}]$ , we have  $v_{i+1} \notin X$ . Observe then that  $X \subseteq C \cup S$ .

We first consider the case when  $v_{i+1} \in N[D^*]$ . Then  $V_{i+1} \setminus N[D^*] = S$  and, as  $H$  is a supergraph of  $G[S \cup C]$  and  $S$  remains an independent set in  $H$ , it follows that  $X$  minimally dominates  $S$  in  $H$ . Note that either  $X$  contains a vertex of  $C$ , and then this vertex dominates  $C$  in  $H$ , or  $X = S$ , and then  $X$  dominates  $C$  in  $H$  as well. We conclude that  $X$  is a minimal dominating set of  $H$ , i.e., that  $X \in \mathcal{D}(H)$  in this case.

We now consider the remaining case when  $v_{i+1} \notin N[D^*]$ ; then  $V_{i+1} \setminus N[D^*] = S \cup \{v_{i+1}\}$ . Observe that now either  $X$  is a minimal dominating set of  $S$ , or there exists  $u \in N(v_{i+1}) \cap X$  such that  $X \setminus \{u\}$  is a dominating set of  $S \setminus N[u]$ , i.e., of  $S \setminus \{u\}$  as

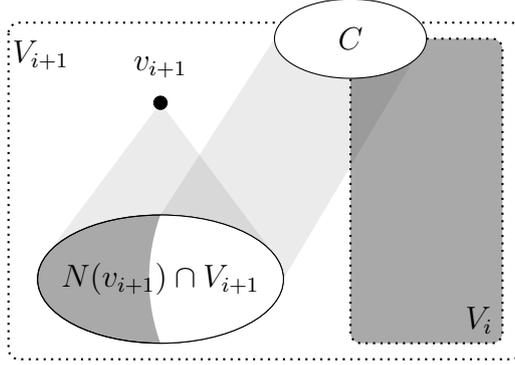


Figure 2: The situation of Lemma 4.2. The set  $N[D^*] \cap V_{i+1}$  is depicted in gray (except  $v_{i+1}$ , in the first case of the proof), and the set  $S = V_{i+1} \setminus (N[D^*] \cup \{v_{i+1}\}) \subseteq N(v_{i+1}) \cap V_{i+1}$  is in white. Note that  $C = N(S) \setminus \{v_{i+1}\}$  may intersect  $V(G) \setminus V_{i+1}$ .

$N(v_{i+1})$  is an independent set. Denote  $D = X$  in the former case and  $D = X \setminus \{u\}$  in the latter case; we now apply a reasoning similar to that from the previous paragraph. Since  $v_{i+1} \notin D$  and  $D$  is a minimal dominating set of  $S$  or  $S \setminus \{u\}$ , it follows that  $D \subseteq C \cup S$ . If  $D \subseteq S$ , then either  $D = S \setminus \{u\}$  (if  $u$  is defined and  $u \in S$ ) or  $D = S$  (otherwise), because  $N(v_{i+1})$  (hence in particular  $S$ ) is an independent set. If  $D \not\subseteq S$ , then  $D \cap C \neq \emptyset$ . In both cases,  $D$  dominates  $C$  in  $H$ , and we conclude that  $X$  (if  $u$  is defined and  $u \in S$ ) or  $D$  (otherwise) is a dominating set of  $H$ . Moreover, since  $\text{Priv}(D, x) \cap S \neq \emptyset$  for each  $x \in D$  it follows that  $X$  or  $D$  is a minimal dominating set of  $H$ . As  $X = D$  or  $X = D \cup \{u\}$  for some vertex  $u$ , we conclude that either  $X$  belongs to  $\mathcal{D}(H)$ , or it belongs to  $\mathcal{D}(H)$  after removing one vertex, a vertex that was here only to dominate  $v_{i+1}$ .

Having considered both cases, the claimed property of elements of  $\mathcal{C}(D^*, i)$  follows. We are left with proving the claimed upper bound on  $|\mathcal{D}(H)|$ . We first show that

$$|\{D \in \mathcal{D}(H) \mid D \cap S = \emptyset\}| \leq (n-1) \cdot |\{D \in \mathcal{D}(H) \mid D \cap S \neq \emptyset\}| + 1. \quad (3)$$

Indeed, consider the map  $f$  that, given  $D \in \{D \in \mathcal{D}(H) \mid D \cap S = \emptyset\}$ ,  $D \neq \emptyset$ , removes one arbitrary vertex from  $D$ , and completes the dominating set by adding all the vertices in the independent set which are no longer dominated. Then,  $f$  maps non-empty elements of  $\{D \in \mathcal{D}(H) \mid D \cap S = \emptyset\}$  to the set  $\{D \in \mathcal{D}(H) \mid D \cap S \neq \emptyset\}$ . Moreover, every element in this second set is the image of at most  $|C| \leq n-1$  elements by  $f$ . This implies the desired bound.

From Inequality (3) we immediately obtain that

$$|\mathcal{D}(H)| \leq n \cdot |\{D \in \mathcal{D}(H) \mid D \cap S \neq \emptyset\}| + 1,$$

so it suffices to prove that

$$|\{D \in \mathcal{D}(H) \mid D \cap S \neq \emptyset\}| \leq |\mathcal{C}(D^*, i)|.$$

To see this, we observe that in fact we have  $\{D \in \mathcal{D}(H) \mid D \cap S \neq \emptyset\} \subseteq \mathcal{C}(D^*, i)$ . Indeed, by Proposition 4.1 we have that every  $D \in \mathcal{D}(H)$  is a minimal dominating set of  $S$  in  $G$ , and it moreover dominates  $v_{i+1}$  provided  $D \cap S \neq \emptyset$ .  $\square$

We now show how to efficiently enumerate the candidate extensions.

**Lemma 4.3.** *There is an algorithm enumerating  $\mathcal{C}(D^*, i)$  in total time  $O(n^4 \cdot |\mathcal{D}(G, A)|)$  and  $O(n^2)$  space whenever  $N(v_{i+1})$  is an independent set.*

*Proof.* First, observe that given any set  $B$  of vertices, we can test in  $O(n^2)$  time and space whether  $B \in \mathcal{C}(D^*, i)$ . Hence, it suffices to enumerate in total time  $O(n^4 \cdot |\mathcal{D}(G, A)|)$  and  $O(n^2)$  space a superset  $\mathcal{F}$  of  $\mathcal{C}(D^*, i)$  of size  $O(n^2 \cdot |\mathcal{D}(G, A)|)$ , and for each element of  $\mathcal{F}$  to test whether it belongs to  $\mathcal{C}(D^*, i)$ . By Lemma 4.2 we can use

$$\mathcal{F} = \{v_{i+1}\} \cup \mathcal{D}(H) \cup \{D \cup \{u\} \mid D \in \mathcal{D}(H), u \in V(G)\},$$

where  $H$  is the split graph defined in the statement of Lemma 4.2. Observe that

$$|\mathcal{F}| \leq (n+1) \cdot |\mathcal{D}(H)| + 1$$

and, using Proposition 4.1, we can enumerate  $\mathcal{F}$  in total time  $O(n^3 \cdot |\mathcal{D}(H)|)$  and space  $O(n^2)$ . It now remains to observe that by Lemma 4.2 and Corollary 3.4 we have

$$|\mathcal{D}(H)| \leq n \cdot |\mathcal{C}(D^*, i)| + 1 \leq n \cdot |\mathcal{D}(G, A)| + 1,$$

so the claimed time complexity follows.  $\square$

We conclude with the following theorem that we state in a more general way than in Section 1, and which is a consequence of Theorem 3.5, Lemma 4.3, and of the fact that when  $G[A]$  is triangle-free, the neighborhood of any vertex is an independent set.

**Theorem 4.4.** *There is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices such that  $G[A]$  is triangle-free, enumerates the set  $\mathcal{D}(G, A)$  in time*

$$O(\text{poly}(n) \cdot |\mathcal{D}(G, A)|^2)$$

and  $O(n^3)$  space.

When  $A = V(G)$ , we have  $\mathcal{D}(G) = \mathcal{D}(G, A)$ . Hence, Theorem 4.4 implies the existence of an algorithm enumerating the minimal dominating sets in triangle-free graphs in output-polynomial time and polynomial space.

## 5 Minimal dominating sets in $K_t$ -free graphs

In this section, we generalize the characterization of Lemma 4.2 and show how to use it to enumerate minimal dominating sets in  $K_t$ -free graphs, at the cost of an increased complexity (see Theorem 5.4).

We start with a general lemma that, roughly, implies that any output-polynomial time algorithm that may repeat outputs can be turned into one without repetition, without increasing space.

**Lemma 5.1.** *Let  $\Sigma_{\text{in}}, \Sigma_{\text{out}}$  be two sets and  $R \subseteq \Sigma_{\text{in}} \times \Sigma_{\text{out}}$  be a relation. Let  $f, s: \Sigma_{\text{in}} \rightarrow \mathbb{N}$  be two functions. Suppose that there is a deterministic algorithm enumerating, given any  $x \in \Sigma_{\text{in}}$ , the set  $\{y \in \Sigma_{\text{out}} \mid xRy\}$  in time at most  $f(x)$  and space at most  $s(x)$ , possibly with repetition. Then there is an algorithm that, on the same input, returns the same output without repetition, in time  $O(f(x)^2)$  and space  $O(s(n))$ .*

*Proof.* Let  $B'$  be the algorithm that on input  $x \in \Sigma_{\text{in}}$  outputs  $\{y \in \Sigma_{\text{out}} \mid xRy\}$ , possibly with repetition, in time at most  $f(x)$  and space at most  $s(x)$ . Elements  $y \in \Sigma_{\text{out}}$  satisfying  $xRy$  will be called *solutions*. We now give an algorithm  $B$  that, on the same input  $x$ , outputs all solutions without repetition. Algorithm  $B$  simulates  $B'$  while counting its

number of output calls. Every time  $B'$  outputs a solution  $y$ ,  $B$  runs a new simulation of  $B'$  to verify whether  $y$  was not output by  $B'$  before. This new simulation is terminated at the first attempt of outputting  $y$ , and for the verification,  $B$  simply checks the output solution counts in both simulations against each other. If  $y$  is indeed output by  $B'$  for the first time, then  $B$  also outputs  $y$ , and otherwise  $B'$  ignores this output and proceeds with the simulation. Thus,  $B$  outputs every solution exactly once: at the first moment when  $B'$  outputs it. The time complexity of  $B$  is  $O(f(x)^2)$ , because for every step of  $B'$  we run a second simulation of  $B'$  that takes time at most  $f(x)$ . The space complexity of  $B$  is at most  $2 \cdot s(x) + O(1) = O(s(x))$ , because we need to store the internal data of two simulations of  $B'$  at any time.  $\square$

By combining Lemma 5.1 and Theorem 3.5, we get the following corollary.

**Corollary 5.2.** *Let  $f: \mathbb{N}^2 \rightarrow \mathbb{N}$  and  $s: \mathbb{N} \rightarrow \mathbb{N}$  be two functions. Suppose that there is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices, a peeling  $(V_0, \dots, V_p)$  of  $G(A)$ ,  $i \in \{0, \dots, p-1\}$  and  $D^* \in \mathcal{D}(G, V_i)$ , enumerates the set  $\mathcal{C}(D^*, i)$  in time at most  $f(n, |\mathcal{D}(G, A)|)$  and space at most  $s(n)$ , possibly with repetition. Then there is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices, enumerates the set  $\mathcal{D}(G, A)$  in time*

$$O(n^4 d^2 + f(n, d)^2 \cdot nd)$$

and space  $O(n \cdot s(n))$ , where  $d = |\mathcal{D}(G, A)|$ .

The aforementioned generalization of Lemma 4.2 is the following.

**Lemma 5.3.** *Let  $G(A)$  be a bicolored graph and  $(V_0, \dots, V_p)$  be a fixed peeling of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ . Let  $i \in \{0, \dots, p-1\}$ ,  $D^* \in \mathcal{D}(G, V_i)$  and  $S = V_{i+1} \setminus (N[D^*] \cup \{v_{i+1}\})$ . Then:*

- if  $v_{i+1} \in N[D^*]$  then  $\mathcal{C}(D^*, i) = \mathcal{D}(G, S)$ ;
- otherwise, every element of  $\mathcal{C}(D^*, i)$  is of the form  $Q \cup \{w\}$  for some  $w \in N[v_{i+1}]$  and  $Q \in \mathcal{D}(G, S \setminus N[w])$ . Furthermore, in this case  $|\mathcal{D}(G, S \setminus N[w])| \leq |\mathcal{C}(D^*, i)|$  for each  $w \in N[v_{i+1}]$ .

*Proof.* By definition,  $\mathcal{C}(D^*, i) = \mathcal{D}(G, V_{i+1} \setminus N[D^*])$ . Note that if  $v_{i+1} \in N[D^*]$  then  $V_{i+1} \setminus N[D^*] = S$ , so we immediately get  $\mathcal{C}(D^*, i) = \mathcal{D}(G, S)$ . This resolves the first case.

Suppose then that  $v_{i+1} \notin N[D^*]$  and consider any  $X \in \mathcal{C}(D^*, i)$ . Since  $X$  minimally dominates  $V_{i+1} \setminus N[D^*] = S \cup \{v_{i+1}\}$ , there exists some  $w \in N[v_{i+1}] \cap X$ . Then  $X \setminus \{w\}$  dominates  $S \setminus N[w]$  and for every element of  $X \setminus \{w\}$ , its private neighbor in  $S \cup \{v_{i+1}\}$  has to actually belong to  $S \setminus N[w]$ . We conclude that  $X \setminus \{w\} \in \mathcal{D}(G, S \setminus N[w])$ , proving the characterization of the elements of  $\mathcal{C}(D^*, i)$  in this case.

We are left with proving the claimed upper bound on  $|\mathcal{D}(G, S \setminus N[w])|$ , for each  $w \in N[v_{i+1}]$ . Take any  $Q \in \mathcal{D}(G, S \setminus N[w])$ ; clearly  $w \notin Q$ . If  $Q$  dominates  $S \cup \{v_{i+1}\}$ , then  $Q$  is also a minimal dominating set of  $S \cup \{v_{i+1}\}$ , because every vertex of  $Q$  has a private neighbor in  $S \setminus N[w] \subseteq S \cup \{v_{i+1}\}$ . Otherwise,  $Q \cup \{w\}$  is a minimal dominating set of  $S \cup \{v_{i+1}\}$ :  $v_{i+1}$  is the private neighbor of  $w$ , and  $w$  could not steal any private neighbors in  $S \setminus N[w]$  from any vertices from  $Q$ . We conclude that either  $Q$  or  $Q \cup \{w\}$  belongs to  $\mathcal{C}(D^*, i)$ , which proves that  $|\mathcal{D}(G, S \setminus N[w])| \leq |\mathcal{C}(D^*, i)|$ .  $\square$

Let us point out the key difference between the statements of Lemma 5.3 and of Lemma 4.2. In Lemma 4.2, we reduced the enumeration of  $\mathcal{C}(D^*, i)$  to the enumeration of  $\mathcal{D}(H)$  for a single split graph  $H$ . In Lemma 5.3, to obtain larger generality we need to separately consider sets  $\mathcal{D}(G, S \setminus N[w])$  for each  $w \in N[v_{i+1}]$ . When enumerating  $\mathcal{C}(D^*, i)$  via enumerating these sets, we will unavoidably obtain repetitions of elements of  $\mathcal{C}(D^*, i)$ . These will be handled using Lemma 5.1 at the cost of an increased complexity.

Observe that by Lemma 5.3, to be able to enumerate the candidate extensions in general (and thus the minimal dominating sets, using Theorem 3.5) in output-polynomial time, it suffices to be able to enumerate the minimal dominating sets of  $G(S)$  and of  $G(S \setminus N[w])$  for every  $w \in N[v_{i+1}]$ . For bicolored graphs  $G(A)$  such that  $G[A]$  is  $K_t$ -free, this can be done by exploiting the fact that  $G[S]$  is  $K_{t-1}$ -free and running the same algorithm on  $G(S)$ , as we shall describe now. We recall that  $\omega(G)$  denotes the size of a largest clique in  $G$ .

**Theorem 5.4.** *There is a function  $p: \mathbb{N} \rightarrow \mathbb{N}$  and an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices, enumerates the set  $\mathcal{D}(G, A)$  in time at most*

$$p(t) \cdot n^{2^{t+1}} \cdot |\mathcal{D}(G, A)|^{2^t}$$

and space at most  $p(t) \cdot n^{t+1}$ , where  $t = \omega(G[A]) + 1$ .

When  $A = V(G)$ , we have  $\mathcal{D}(G) = \mathcal{D}(G, A)$ . Hence, Theorem 5.4 implies the existence of an algorithm enumerating, for every integer  $t \geq 1$ , the minimal dominating sets in  $K_t$ -free graphs in output-polynomial time and polynomial space. We stress that we provide a single algorithm for all values of  $t$ —note that as stated, the algorithm does not require knowledge of  $t$ .

*Proof of Theorem 5.4.* In this proof we consider two algorithms A and B that recursively call each other in order to enumerate the minimal dominating sets of a bicolored graph. We first give their specifications, then describe them, and finally prove that they perform as specified. Let  $f: \mathbb{N}^3 \rightarrow \mathbb{N}$  be defined by  $f(n, d, t) = n^{2^{t+1}-3} \cdot d^{2^t-1}$ , for every  $n, d, t \in \mathbb{N}$ .

**Specifications of A and B** We will show that the algorithms A and B have the following properties  $P$  and  $Q$ , for every  $t \geq 1$  in case of  $P(t)$  and every  $t \geq 2$  in case of  $Q(t)$ .

$P(t)$ : There is a constant  $p(t) \in \mathbb{N}$  such that given an  $n$ -vertex graph  $G$  and a set  $A \subseteq V(G)$  such that  $G[A]$  is  $K_t$ -free, A outputs  $\mathcal{D}(G, A)$  in time at most  $p(t) \cdot f(n, |\mathcal{D}(G, A)|, t)$  and space at most  $p(t) \cdot n^{t+1}$ .

$Q(t)$ : There is a constant  $q(t) \in \mathbb{N}$  such that given a  $n$ -vertex graph  $G$ , a set  $A \subseteq V(G)$  such that  $G[A]$  is  $K_t$ -free, a peeling  $(V_0, \dots, V_p)$  of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ ,  $i \in \{0, \dots, p-1\}$ , and  $D^* \in \mathcal{D}(G, V_i)$ , B outputs  $\mathcal{C}(D^*, i)$  in time at most  $q(t) \cdot n^2 \cdot f(n, |\mathcal{D}(G, A)|, t-1)^2$  and space at most  $q(t) \cdot n^t$ .

The statement of Theorem 5.4 is implied by  $P(t)$  holding for all  $t \geq 1$ . In order to prove it, we will also show that  $Q(t)$  holds for every  $t \geq 2$ . Let us first describe A.

**Description of A** The algorithm A is the one given by Theorem 3.5 that takes as input a bicolored graph  $G(A)$ , using B as a routine to enumerate candidate extensions. We will show below that B indeed does so.

**Description of B** Recall that B takes as input a bicolored graph  $G(A)$ , a peeling  $(V_0, \dots, V_p)$  of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ , an integer  $i \in \{0, \dots, p-1\}$ , and a set  $D^* \in \mathcal{D}(G, V_i)$ .

We first describe an auxiliary routine B'. Let  $S = V_{i+1} \setminus (N[D^*] \cup \{v_{i+1}\})$ . Lemma 5.3 above allows us to consider two cases depending on whether  $D^*$  dominates  $v_{i+1}$  or not:

- (i) if  $v_{i+1} \in N[D^*]$ , we call algorithm A on  $G(S)$  to enumerate  $\mathcal{D}(G, S)$  and we give the same output;
- (ii) otherwise, we iterate over all  $w \in N[v_{i+1}]$  and  $Q \in \mathcal{D}(G, S \setminus N[w])$  (where the latter is obtained via a call to A) and output  $Q \cup \{w\}$  if and only if it is a candidate extension of  $D^*$ .

We are now done with B'. As we will show later, B' enumerates  $\mathcal{C}(D^*, i)$ , however each element may be repeated, up to  $n$  times. Then B is obtained from B' using Lemma 5.1. This concludes the description of B.

**Correctness of A and B** Now that we described the algorithms A and B, we show that they conform to their specifications, i.e., we prove that  $P(t)$  holds for every  $t \geq 1$  and that  $Q(t)$  holds for every  $t \geq 2$ . The proof by induction on  $t$  is split in lemmas.

**Lemma 5.5.**  $P(1)$  holds.

*Proof.* The statement  $P(1)$  deals with pairs  $(G, A)$  such that  $G[A]$  is  $K_1$ -free, so  $A = \emptyset$ . In these cases we clearly have  $\mathcal{D}(G, A) = \{\emptyset\}$ . Notice that such inputs are correctly handled by algorithm A. Checking whether  $A$  is empty and returning  $\{\emptyset\}$  takes  $O(n)$  time and  $O(n^2)$  space. We define  $p(1)$  as an integer such that these steps take at most  $p(1) \cdot n$  time and at most  $p(1) \cdot n^2$  space on an input graph of order  $n$ . As  $f(n, |\mathcal{D}(G, A)|, t) = n$  in this case,  $P(1)$  holds.  $\square$

**Lemma 5.6.** For every integer  $t \geq 1$ ,  $P(t) \Rightarrow Q(t+1)$ .

*Proof.* Let  $t \geq 1$  and let us assume that the statement  $P(t)$  holds (in particular,  $p(t)$  is defined). Let  $\mathcal{I} = (G, A, V_0, \dots, V_p, v_1, \dots, v_p, i, D^*)$  be an input of B such that  $G[A]$  is  $K_{t+1}$ -free. Let us define  $n = |G|$  and  $d = |\mathcal{D}(G, A)|$ . We review the description of B to show that  $Q(t+1)$  holds. We first consider the auxiliary routine B'.

**Claim 5.7.** Given  $\mathcal{I}$ , the algorithm B' enumerates  $\mathcal{C}(D^*, i)$  with each output repeated up to  $n$  times, in time at most  $k \cdot n \cdot f(n, d, t)$  and space at most  $k \cdot n^{t+1}$ , for some constant  $k$ .

*Proof.* Let  $S = V_{i+1} \setminus (\{v_{i+1}\} \cup N[D^*])$ . Note that as  $D^*$  dominates  $V_i$ , we have  $S \subseteq N(v_{i+1}) \cap V_{i+1}$ . Also,  $S$  can be computed in  $O(n^2)$  time and space.

Since  $i < p$ , we have  $V_{i+1} \subseteq A$ , from the definition of a peeling. In particular,  $G[V_{i+1}]$  is  $K_{t+1}$ -free. As  $S \subseteq N(v_{i+1}) \cap V_{i+1}$ , we get that  $G[S]$  is  $K_t$ -free. Hence, when applying the algorithm recursively to enumerate  $\mathcal{D}(G, S')$  for any  $S' \subseteq S$ , we may use the already established property  $P(t)$ , yielding the following:

*Remark 5.8.* For any  $S' \subseteq S$ , a call to A on  $(G, S')$  returns  $\mathcal{D}(G, S')$  in time at most  $p(t) \cdot f(n, |\mathcal{D}(G, S')|, t)$  and space at most  $p(t) \cdot n^{t+1}$ .

If  $v_{i+1} \in N[D^*]$ , then, by Lemma 5.3, we enumerate  $\mathcal{C}(D^*, i)$  without repetitions simply by enumerating  $\mathcal{D}(G, S)$ . By Remark 5.8, this takes time

$$\begin{aligned} & p(t) \cdot f(n, |\mathcal{D}(G, S)|, t) \\ &= p(t) \cdot f(n, |\mathcal{C}(D^*, i)|, t) && \text{(by Lemma 5.3)} \\ &\leq p(t) \cdot f(n, d, t) && \text{(by Corollary 3.4)} \end{aligned}$$

and space at most  $p(t) \cdot n^{t+1}$ .

Now suppose that  $v_{i+1} \notin N[D^*]$ . Then, by Lemma 5.3, we enumerate all elements of the set  $\mathcal{C}(D^*, i)$ , however each of them is enumerated once per every form  $Q \cup \{w\}$  it can take, where  $w \in N[v_{i+1}]$  and  $Q \in \mathcal{D}(G, S \setminus N[w])$ . Every such occurrence is characterized by the choice of  $w$ , hence there are at most  $n$  of them and, consequently, every member of  $\mathcal{C}(D^*, i)$  is enumerated at most  $n$  times.

Regarding time and space complexity we perform at most  $n$  times (once for every choice of  $w \in N[v_{i+1}]$ ) the following operations:

- constructing  $S \setminus N[w]$ , in  $O(n)$  time and space;
- calling **A** on  $(G, S \setminus N[w])$ , in time at most  $p(t) \cdot f(n, |\mathcal{D}(G, S \setminus N[w])|, t)$  and space at most  $p(t) \cdot n^{t+1}$ , by Remark 5.8;
- checking, for each set  $Q$  among the outputs of **A**, whether  $Q \cup \{w\}$  belongs to  $\mathcal{C}(D^*, i)$ , in  $O(n^2)$  time and space. There are at most  $d$  outputs.

By Lemma 5.3 and Corollary 3.4, we have

$$|\mathcal{D}(G, S \setminus N[w])| \leq |\mathcal{C}(D^*, i)| \leq d.$$

Therefore, in total the time complexity of these steps adds up to:

$$\begin{aligned} & n \cdot [O(n) + p(t) \cdot f(n, d, t) + O(n^2 \cdot d)] \\ &= O(p(t) \cdot n \cdot f(n, d, t)) && \text{(as } t \geq 2 \text{ in this case).} \end{aligned} \quad (4)$$

Similarly, the space complexity can be upper-bounded by  $O(p(t) \cdot n^{t+1})$ . We conclude by setting  $k$  as  $p(t)$  times the maximum of the constants hidden in the  $O(\cdot)$  notation above.  $\square$

As proved in Lemma 5.1, the algorithm of Claim 5.7 can be turned into an algorithm **B** that does not repeat outputs. That is, there is a constant  $q(t+1)$  (depending on  $k$ ) such that given  $\mathcal{I}$ , **B** runs in time at most  $q(t+1) \cdot n^2 \cdot f(n, d, t)^2$  and space at most  $q(t+1) \cdot n^{t+1}$ . Hence  $Q(t+1)$  holds, as desired.  $\square$

**Lemma 5.9.** *For every integer  $t \geq 2$ ,  $Q(t) \Rightarrow P(t)$ .*

*Proof.* Let us assume that for some integer  $t \geq 2$ , the statement  $Q(t)$  holds (and in particular  $q(t)$  is defined). Let  $G$  be a graph and  $A \subseteq V(G)$  be such that  $G[A]$  is  $K_t$ -free. We set  $n = |G|$  and  $d = |\mathcal{D}(G, A)|$ . By  $Q(t)$ , the enumeration of candidate extensions in  $G(A)$  can be carried out by **B** in total time at most

$$q(t) \cdot n^2 \cdot f(n, d, t-1)^2$$

and space at most  $q(t) \cdot n^t$ . According to Theorem 3.5, A then enumerates  $\mathcal{D}(G, A)$  in time

$$\begin{aligned} & O(n^4 \cdot d^2 + q(t) \cdot n^3 \cdot f(n, d, t - 1)^2 \cdot d) \\ = & O(n^4 \cdot d^2 + q(t) \cdot f(n, d, t)) && \text{(by the definition of } f) \\ = & O(q(t) \cdot f(n, d, t)) && \text{(as } t \geq 2) \end{aligned}$$

and space  $O(q(t) \cdot n^{t+1})$ . Therefore, there is a constant  $p(t)$  (depending on  $q(t)$ ) such that A runs on this input in time at most  $p(t) \cdot f(n, d, t)$  and space at most  $p(t) \cdot n^{t+1}$ . This proves  $P(t)$ .  $\square$

**Concluding the proof.** We proceed by induction on  $t$ . The base case  $P(1)$  follows from Lemma 5.5. The induction step that, for every integer  $t \geq 1$ ,  $P(t)$  implies  $P(t + 1)$ , is obtained by combining Lemmas 5.6 and 5.9. We conclude that  $P(t)$  holds for every integer  $t \geq 1$ . That is, the algorithm A has the properties claimed in the statement of the theorem.  $\square$

We note that the complexity of the algorithm of Theorem 5.4 for  $K_t$ -free graphs could be slightly improved when  $t \geq 3$ , using Theorem 4.4 as a base case, however that would not remove the exponential contribution of  $t$  to the degree of the polynomial.

## 6 Variants of $K_t$ -free graphs

We give output-polynomial time algorithms for classes related to  $K_t$ -free graphs relying on the algorithms and characterizations of candidate extensions given in Sections 3, 4, and 5.

### 6.1 Forbidding $K_t + K_2$

In this section we show how the algorithm of Theorem 5.4 on  $K_t$ -free graphs can be extended to the setting of  $(K_t + K_2)$ -free graphs.

**Theorem 6.1.** *There is an algorithm that, for every fixed  $t \in \mathbb{N}$ , enumerates minimal dominating sets in  $(K_t + K_2)$ -free graphs in output-polynomial time and polynomial space.*

*Proof.* Let  $t \in \mathbb{N}$  and let  $G$  be a  $(K_t + K_2)$ -free graph. It is well-known that the minimal dominating sets of  $G$  that induce edgeless subgraphs are exactly the maximal independent sets of  $G$ . We can therefore enumerate these using the polynomial delay algorithms of Tsukiyama et al. [TIAS77] for maximal independent sets. In the sequel we may thus focus on those minimal dominating sets of  $G$  that induce at least one edge.

We show how to enumerate, for every edge  $uv$  of  $G$ , the minimal dominating sets of  $G$  that contain both  $u$  and  $v$ . Let  $A_{uv} = V(G) \setminus N[\{u, v\}]$  and observe that  $G[A_{uv}]$  is  $K_t$ -free. First, we enumerate  $G(A_{uv})$  using the algorithm of Theorem 5.4, which runs in output-polynomial time and polynomial space, as  $t$  is fixed. For every  $D \in \mathcal{D}(G, A_{uv})$  obtained from the aforementioned call, we output  $D \cup \{u, v\}$  if it is a minimal dominating set of  $G$ , and discard  $D$  otherwise. By Lemma 3.3 (applied for  $H = G$  and  $B = V(G)$ ) we have  $|\mathcal{D}(G, A_{uv})| \leq |\mathcal{D}(G)|$ . Hence, enumerating  $\mathcal{D}(G, A_{uv})$  produces all those minimal dominating sets of  $G$  that at least induce the edge  $uv$  in time  $\text{poly}(n \cdot |\mathcal{D}(G)|)$  and space  $\text{poly}(n)$ , where the degrees of these polynomials depend on  $t$  (see Theorem 5.4).

Now that we know how to enumerate minimal dominating sets that induce at least one particular edge, we can run the above routine for every edge of  $G$  to enumerate all minimal dominating sets of  $G$ , possibly with repetitions. Observe that the same output can be repeated at most  $|E(G)|$  times. Then, repetitions are avoided using Lemma 5.1 with  $\Sigma_{\text{in}}$  being the set of all graphs,  $\Sigma_{\text{out}}$  the set of all vertex sets, and  $R$  the relation that associates every graph to its minimal dominating sets.  $\square$

## 6.2 Forbidding $K_t - e$

Another interesting case is the one of  $(K_t - e)$ -free graphs. In this section we show how the characterization of Lemma 5.3 can be used to enumerate candidate extensions in diamond-free graphs (which are  $(K_t - e)$ -free for  $t = 4$ ), which by Theorem 3.5 gives an output-polynomial time algorithm enumerating minimal dominating sets in this class. We leave open the existence of such an algorithm in the case when  $t \geq 5$ .

In what follows, we consider a bicolored graph  $G(A)$  on  $n$  vertices such that  $G$  is diamond-free, together with a fixed peeling  $(V_0, \dots, V_p)$  of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ . Then, we consider

$$i \in \{0, \dots, p-1\}, \quad D^* \in \mathcal{D}(G, V_i),$$

and define  $S = V_{i+1} \setminus (N[D^*] \cup \{v_{i+1}\})$  and  $\mathcal{C}(D^*, i)$  as in Sections 3, 4, and 5. Note that contrarily to the triangle-free case and the  $K_t$ -free case, we here require the whole graph  $G$  to be diamond-free and not only  $G[A]$ . We start with an easy observation.

**Observation 6.2.** *For every vertex  $u$  of  $G$ ,  $G[N(u)]$  is  $P_3$ -free. Then  $G[N(v_{i+1})]$ , hence also  $G[S]$ , can be partitioned into a disjoint union of cliques (i.e., it is a cluster graph).*

We will show how to minimally dominate one clique of  $S$ , then a disjoint union of cliques of  $S$ , and will conclude with the enumeration of  $\mathcal{C}(D^*, i)$ .

**Lemma 6.3.** *Let  $K$  be a clique of  $G[S]$  and  $u$  be a vertex in  $G - S$ ,  $u \neq v_{i+1}$ , that is adjacent to some vertex of  $K$ . If  $u$  is adjacent to  $v_{i+1}$ , then it is complete to  $K$ . Otherwise  $u$  has exactly one neighbor in  $K$ .*

*Proof.* If  $u \in N(v_{i+1})$  then, as  $G[N(v_{i+1})]$  is  $P_3$ -free and  $K \subseteq N(v_{i+1})$ ,  $u$  is complete to  $K$ . If  $u$  is not adjacent to  $v_{i+1}$ , then it has exactly one neighbor in  $K$ , as otherwise  $\{a, b, u, v_{i+1}\}$  would induce a diamond in  $G$ , for any two neighbors  $a, b \in K$  of  $u$ .  $\square$

**Lemma 6.4.** *Let  $K$  be a clique in  $G[S]$ . Then  $\mathcal{D}(G, K)$  can be enumerated in total time  $O(n^2 + n \cdot |\mathcal{D}(G, K)|)$  and  $O(n^2)$  space.*

*Proof.* We describe an algorithm enumerating  $\mathcal{D}(G, K)$  in the specified time and space bounds. We first output  $\{v_{i+1}\}$  as it is complete to  $K$ . We then output all vertices  $u \in N(v_{i+1})$  such that  $u \in K$  or  $u$  is adjacent to some vertex of  $K$ . By Lemma 6.3, these vertices are also complete to  $K$ . Then, for every  $x \in K$ , we compute the neighborhood of  $x$  outside of  $N(v_{i+1})$  in total time  $O(n^2)$ . By Lemma 6.3, these neighborhoods are disjoint. At last, we enumerate the unordered Cartesian products of these neighborhoods. This can clearly be done in total time of  $O(n \cdot |\mathcal{D}(G, K)|)$  using  $O(n)$  space as they are disjoint. Clearly, every element in such an unordered Cartesian product is a minimal dominating set of  $K$ , and the described algorithm performs within the specified time and space bounds. The correctness of the algorithm follows from Lemma 6.3.  $\square$

**Lemma 6.5.** *Let  $W$  be a subset of  $S$ . Then  $\mathcal{D}(G, W)$  can be enumerated in total time  $O(n^7 \cdot |\mathcal{D}(G, A)|^3)$  and  $O(n^3)$  space.*

*Proof.* We use the ordered generation described in Section 3. The algorithm first computes a peeling  $(U_1, \dots, U_q)$  of  $G(W)$  with vertex sequence  $(u_1, \dots, u_q)$ , in  $O(n^2)$  time and space. Note that  $N[u_1] \cap W, \dots, N[u_q] \cap W$  is exactly the disjoint clique partition of  $G[W]$ ; denote these sets by  $W_1, \dots, W_q$ . Given  $j \in \{0, \dots, q-1\}$  and  $D^\circ \in \mathcal{D}(G, U_j)$ , we define  $\mathcal{C}'(D^\circ, j)$  as the set of candidate extensions of  $(D^\circ, j)$  with respect to the chosen peeling of  $G(W)$  and we show how to enumerate  $\mathcal{C}'(D^\circ, j)$  in time  $O(n^6 \cdot |\mathcal{D}(G, A)|^2)$  and using  $O(n^2)$  space.

We rely on the same characterization of candidate extensions that we use in the proof of Theorem 5.4, i.e., Lemma 5.3. Recall that this lemma allows us to consider two cases depending on whether  $D^\circ$  dominates  $u_{j+1}$  or not. Let  $Y = W_{j+1} \setminus N[D^\circ]$ .

If  $D^\circ$  dominates  $u_{j+1}$ , then we can enumerate  $\mathcal{C}'(D^\circ, j)$  by just enumerating  $\mathcal{D}(G, Y)$ , as these sets coincide. As  $Y$  is a clique in  $G[S]$ , by Lemma 6.4 we can enumerate  $\mathcal{D}(G, Y)$  in time  $O(n^2 + n \cdot |\mathcal{D}(G, Y)|)$  and space  $O(n^2)$ . By Lemma 3.3 we have  $|\mathcal{D}(G, Y)| \leq |\mathcal{D}(G, A)|$ , hence the procedure runs within the required time and space complexity.

In the remaining case when  $D^\circ$  does not dominate  $u_{j+1}$ , we iterate over all  $w \in N[u_{j+1}]$  and  $Q \in \mathcal{D}(G, Y \setminus N[w])$  (obtained via a call to the algorithm of Lemma 6.4) and output  $Q \cup \{w\}$  if and only if this set belongs to  $\mathcal{C}'(D^\circ, j)$ , which can be checked in  $O(n^2)$  time and space. Again, by Lemma 3.3 we have  $|\mathcal{D}(G, Y \setminus N[w])| \leq |\mathcal{D}(G, A)|$  for all  $w$  as above. Hence, in total, the described algorithm enumerates  $\mathcal{C}'(D^\circ, i)$ , possibly with repetitions, in time  $O(n^3 \cdot |\mathcal{D}(G, A)|)$  and using  $O(n^2)$  space. Using Corollary 5.2, we obtain an algorithm enumerating  $\mathcal{D}(G, W)$  in time  $O(n^7 \cdot |\mathcal{D}(G, A)|^3)$ , and using  $O(n^3)$  space.  $\square$

**Lemma 6.6.** *There is an algorithm enumerating  $\mathcal{C}(D^*, i)$ , possibly with repetitions, in total time  $O(n^8 \cdot |\mathcal{D}(G, A)|^3)$  and using  $O(n^3)$  space.*

*Proof.* We apply the same argument as in the previous lemma, and in the proof of Theorem 5.4. Lemma 5.3 allows us to consider two cases depending on whether  $D^*$  dominates  $v_{i+1}$  or not. If  $D^*$  dominates  $v_{i+1}$ , we call the algorithm of Lemma 6.5 to enumerate  $\mathcal{D}(G, S)$  without repetitions in total time  $O(n^7 \cdot |\mathcal{D}(G, A)|^3)$  and  $O(n^3)$  space. If  $D^*$  does not dominate  $v_{i+1}$ , we iterate over all  $w \in N[v_{i+1}]$  and  $Q \in \mathcal{D}(G, S \setminus N[w])$  (obtained via a call to the algorithm of Lemma 6.5 as  $S \setminus N[w] \subseteq S$ ) and output  $Q \cup \{w\}$  if and only if this set belongs to  $\mathcal{C}(D^*, i)$ . An analogous complexity analysis shows that this algorithm runs in time  $O(n^8 \cdot |\mathcal{D}(G, A)|^3)$  and uses  $O(n^3)$  space, and it enumerates  $\mathcal{C}(D^*, i)$  possibly with repetitions.  $\square$

As a consequence of Corollary 5.2 and Lemma 6.6, we get the following.

**Theorem 6.7.** *There is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices such that  $G$  is diamond-free, enumerates the set  $\mathcal{D}(G, A)$  in time*

$$O(\text{poly}(n) \cdot |\mathcal{D}(G, A)|^7)$$

*and  $O(n^4)$  space.*

Note that when  $A = V(G)$ , we have  $\mathcal{D}(G) = \mathcal{D}(G, A)$ . Hence, Theorem 6.7 implies the existence of an algorithm enumerating the minimal dominating sets in diamond-free graphs in output-polynomial time and using polynomial space, which is one of the two cases of Theorem 1.2.

### 6.3 Paw-free graphs

We now consider the exclusion of a specific graph, the paw, and show that DOM-ENUM admits an output-polynomial time algorithm in paw-free graphs.

In what follows, we consider a bicolored graph  $G(A)$  on  $n$  vertices such that  $G$  is paw-free, together with a fixed peeling  $(V_0, \dots, V_p)$  of  $G(A)$  with vertex sequence  $(v_1, \dots, v_p)$ . Then, we consider

$$i \in \{0, \dots, p-1\}, \quad D^* \in \mathcal{D}(G, V_i),$$

and define  $S = V_{i+1} \setminus (N[D^*] \cup \{v_{i+1}\})$  and  $\mathcal{C}(D^*, i)$  as in Sections 3, 4 and 5. As in the previous section we stress that we require the whole graph  $G$  to be paw-free, and not only  $G[A]$ . We start with an easy observation.

**Observation 6.8.** *For every vertex  $u$  of  $G$ ,  $G[N(u)]$  is  $\overline{P}_3$ -free. Hence  $G[S]$  is a complete multipartite graph (also called a co-cluster graph).*

Note that when enumerating  $\mathcal{C}(D^*, i)$  (i.e., the minimal dominating sets of  $G(S)$ ), we may safely ignore the edges between two vertices of  $G - S$ . Therefore, if  $S$  is an independent set, we can delete all edges of  $G - S$  (in  $O(n^2)$  time) to obtain that  $N(v_{i+1})$  is an independent set and then apply the algorithm enumerating  $\mathcal{C}(D^*, i)$  in this setting given by Lemma 4.3. In the next lemma, we consider the case where  $S$  contains at least one edge.

**Lemma 6.9.** *Assume that  $G[S]$  contains at least one edge, and let  $u$  be a vertex of  $G$ ,  $u \neq v_{i+1}$ , that has a neighbor in  $S$ . If  $u$  is not adjacent to  $v_{i+1}$ , then it is complete to  $S$ . Otherwise,  $u$  is complete to  $S \setminus I_j$ , for some  $j \in \{1, \dots, q\}$  where  $I_1, \dots, I_q$  is the complete multipartition of  $G[S]$  (every  $I_j$  induces an independent set in  $G$ , while vertices from different  $I_j$ 's are adjacent).*

*Proof.* As by assumption  $G[S]$  contains an edge, we have  $q \geq 2$ . Assume first that  $u$  is not adjacent to  $v_{i+1}$ , but has a neighbor in  $S$ ; in particular  $u \notin S$ . Suppose for contradiction that  $u$  is not complete to  $S$ . Hence there are vertices  $x \in S \cap N(u)$  and  $y \in S \setminus N(u)$ . Note that  $xy \notin E(G)$  as otherwise  $\{u, v_{i+1}, x, y\}$  induces a paw in  $G$ . Then  $x, y \in I_j$  for some  $j \in \{1, \dots, q\}$ . Let  $z \in S \setminus I_j$ ; such a vertex exists as  $q \geq 2$  and it is complete to  $\{v_{i+1}, x, y\}$  by definition of the  $I_k$ 's. Then, either  $uz \in E(G)$  and  $\{u, x, y, z\}$  induces a paw, or  $uz \notin E(G)$  and  $\{u, v_{i+1}, y, z\}$  does, a contradiction.

Assume now that  $u$  is adjacent to  $v_{i+1}$ . If  $u$  belongs to  $S$ , then it belongs to some  $I_j$ ,  $j \in \{1, \dots, q\}$  and is complete to  $S \setminus I_j$ , by definition of the  $I_k$ 's. We now assume  $u \in N(v_{i+1}) \setminus S$ . If there is no  $j \in \{1, \dots, q\}$  such that  $u$  is complete to  $S \setminus I_j$ , then, as  $q \geq 2$ ,  $u$  has at least two non-neighbors  $x \in I_{j'}$  and  $y \in I_{j''}$  for two different  $j', j'' \in \{1, \dots, q\}$ . Then  $\{u, v_{i+1}, x, y\}$  induces a paw in  $G$ , a contradiction.  $\square$

**Lemma 6.10.** *There is an algorithm enumerating  $\mathcal{C}(D^*, i)$  in total time  $O(n^5 \cdot |\mathcal{D}(G, A)|)$  and  $O(n^2)$  space.*

*Proof.* In the case where  $S$  induces an independent set, we use the algorithm of Lemma 4.3 to enumerate  $\mathcal{C}(D^*, i)$  in time

$$O(n^4 \cdot |\mathcal{D}(G, A)|)$$

and  $O(n^2)$  space. Otherwise, we deduce from Lemma 6.9 that minimal dominating sets of  $S$  are either of size at most two, or of the form  $I_j$  for some  $j \in \{1, \dots, q\}$ . If  $v_{i+1} \in N[D^*]$ ,

that is if  $S = V_{i+1} \setminus N[D^*]$ , we try each of these sets and output those that minimally dominate  $S$ ; this can be done in total time  $O(n^4)$ . This enumerates  $\mathcal{C}(D^*, i)$  by definition. If  $v_{i+1} \notin N[D^*]$ , we first output  $I_j$  for every  $j \in \{1, \dots, q\}$ . Then, we iterate over all vertex subsets of size at most three and output those that minimally dominate  $S$ ; this can be done in total time  $O(n^5)$ . This will enumerate  $\mathcal{C}(D^*, i)$ , for the following reason implied by Lemma 6.9. If  $X \in \mathcal{C}(D^*, i)$ , then either  $X = I_j$  for some  $j \in \{1, \dots, q\}$ , or  $X$  contains at most three vertices: one with  $v_{i+1}$  as a private neighbor and at most 2 with private neighbors in  $S$ .  $\square$

As a consequence of Theorem 3.5 and Lemma 6.10, we get the following.

**Theorem 6.11.** *There is an algorithm that, given a bicolored graph  $G(A)$  on  $n$  vertices such that  $G$  is paw-free, enumerates the set  $\mathcal{D}(G, A)$  in time*

$$O(\text{poly}(n) \cdot |\mathcal{D}(G, A)|^2)$$

and  $O(n^3)$  space.

Note that when  $A = V(G)$ , we have  $\mathcal{D}(G) = \mathcal{D}(G, A)$ . Hence, Theorem 6.11 implies the existence of an algorithm enumerating the minimal dominating sets in paw-free graphs in output-polynomial time and using polynomial space, which is the second of the two cases of Theorem 1.2.

## 7 Technique limitations

In this section, we discuss various obstacles that we detected in our attempts to improve our results or proofs.

### 7.1 A standard technique fails for bipartite graphs

A natural technique (sometimes called *flashlight search* or *backtrack*) to enumerate valid solutions to a given problem such as, for instance, sets of vertices satisfying a given property, is to build them element by element. If during the construction one detects that the current partial solution cannot be extended into a valid one, then it can be discarded along with all the other partial solutions that contain it. Note that in order to apply this technique, one should be able to decide whether a given partial solution can be completed into a valid one. It turns out that for minimal dominating sets, this problem (called the *extension problem*) is NP-complete [KLMN11], even when restricted to split graphs [KLM<sup>+</sup>15]. We show that it remains NP-complete in bipartite graphs, so in particular on  $(K_t + K_2)$ -free graphs for every  $t \geq 3$ . This stands in contrast with Theorem 1.1 and suggests that, indeed, a more involved technique was needed to obtain our results.

The extension problem, denoted DCS, is formally defined as follows. Given a graph  $G$  and a set  $A \subseteq V(G)$  of vertices, is there a minimal dominating set  $D \in \mathcal{D}(G)$  such that  $A \subseteq D$ .

**Theorem 7.1.** *DCS restricted to bipartite graphs is NP-complete.*

*Proof.* Since DCS is NP-complete in the general case, it is clear that DCS is in NP even when restricted to bipartite graphs. Let us now present a hardness reduction from SAT.

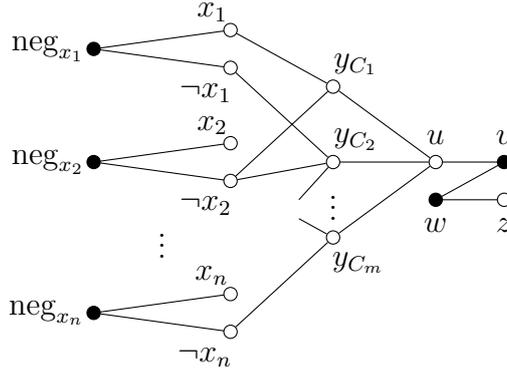


Figure 3: A bipartite graph  $G$  and a set  $A \subseteq V(G)$  constructed from an instance of SAT with variables  $x_1, \dots, x_n$  and clauses  $C_1, \dots, C_m$ . Black vertices constitute the set  $A$ . Then  $A$  can be extended into a minimal dominating set  $D$  of  $G$  if and only if there is a truth assignment of the variable satisfying all the clauses.

Given an instance  $\varphi$  of SAT with variables  $x_1, \dots, x_n$  and clauses  $C_1, \dots, C_m$ , we construct a bipartite graph  $G$  and a set  $A \subseteq V(G)$  such that there exists a minimal dominating set containing  $A$  if and only if there exists a truth assignment to the variables of  $\varphi$  that satisfies all the clauses. The graph  $G$  has vertex bipartition  $(X, Y)$ , defined as follows.

The first part  $X$  contains two special vertices  $u$  and  $w$ , and for every variable  $x_i$ , one vertex for each of the literals  $x_i$  and  $\neg x_i$ . The second part  $Y$  contains one vertex  $y_{C_j}$  per clause  $C_j$ , one vertex  $\text{neg}_{x_i}$  per variable  $x_i$ , and two special vertices  $v$  and  $z$ . For every  $i \in \{1, \dots, n\}$  we make  $\text{neg}_{x_i}$  adjacent to the two literals  $x_i$  and  $\neg x_i$  and for every  $j \in \{1, \dots, m\}$  we make  $y_{C_j}$  adjacent to  $u$  and to every literal  $C_j$  contains. Finally, we add edges to form the path  $uvwz$  and set  $A = \{\text{neg}_{x_1}, \dots, \text{neg}_{x_n}, v, w\}$ . Clearly this graph can be constructed in polynomial time from  $\varphi$ . The construction is illustrated in Figure 3.

Let us show that  $A$  can be extended into a minimal dominating set of  $G$  if and only if  $\varphi$  has a truth assignment that satisfies all the clauses. The proof is split into two claims. A *partial assignment* of  $\varphi$  is a truth assignment of a subset of the variables  $x_1, \dots, x_n$ . Observe that a partial assignment may satisfy all the clauses (i.e., the values of the non-assigned variables do not matter). A partial assignment that satisfies all the clauses is called a *minimal assignment* if none of its proper subsets satisfies all the clauses.

**Claim 7.2.** *Let  $S \subseteq \{x_1, \neg x_1, \dots, x_n, \neg x_n\}$  be a set containing at most one literal for each variable. Then  $S$  minimally dominates  $\{y_{C_1}, \dots, y_{C_m}\}$  if and only if its elements form a minimal assignment of  $\varphi$ .*

*Proof.* Suppose  $S$  is as above and  $S$  minimally dominates  $\{y_{C_1}, \dots, y_{C_m}\}$ . Consider any  $j \in \{1, \dots, m\}$ . Since  $y_{C_j} \notin S$ , the set  $S$  contains a neighbor  $x$  of  $y_{C_j}$ . By construction,  $x$  is a literal appearing in  $C_j$ . Hence, the literals present in  $S$  form a partial assignment of the variables of  $\varphi$  satisfying all its clauses. Moreover, this partial assignment is minimal by the minimality of  $S$ . The proof in the other direction is analogous.  $\square$

**Claim 7.3.** *If  $D$  is a minimal dominating set of  $G$  containing  $A$ , then  $D \setminus A \subseteq \{x_1, \neg x_1, \dots, x_n, \neg x_n\}$  and  $D$  contains at most one literal of each variable.*

*Proof.* Notice that  $\text{Priv}(A, v) = \{u\}$ . If  $y_{C_j}$  belongs to  $D$  for some  $j \in \{1, \dots, m\}$ , then  $\text{Priv}(D, v) = \emptyset$ , a contradiction to the minimality of  $D$ . For similar reasons  $u, z \notin D$ .

Hence  $D \cap \{u, z, y_{C_1}, \dots, y_{C_m}\} = \emptyset$ . Besides, for every  $i \in \{1, \dots, m\}$ ,  $D$  contains at most one of  $x_i$  and  $\neg x_i$ , as otherwise  $\text{Priv}(D, \text{neg}_{x_i})$  would be empty, again contradicting the minimality of  $D$ . This proves the claim.  $\square$

If  $A$  can be extended into a minimal dominating set  $D$  of  $G$ , then by combining the two claims above, we deduce that  $\varphi$  has a truth assignment that satisfies all clauses. Conversely, if  $\varphi$  has a truth assignment satisfying all the clauses, then it also has a minimal truth assignment satisfying all the clauses, so there is a set  $S$  as in the statement of Claim 7.2. In  $S \cup A$ , every element of  $S$  has a private neighbor, as a consequence of the minimality of  $S$  and the fact that no element of  $A$  has a neighbor among the clause vertices. Besides, each of  $\text{neg}_{x_1}, \dots, \text{neg}_{x_n}$  has a private neighbor (because  $S$  contains at most one of the two literals for each variable) and it is easy to see that the same holds for  $v$  and  $w$ . Hence  $S \cup A$  is a minimal dominating set of  $G$ .

Given an instance  $\varphi$  of SAT, we constructed in polynomial time an instance  $(G, A)$  of DCS that is equivalent to satisfiability of  $\varphi$ . This proves that DCS is NP-hard.  $\square$

## 7.2 Limitations of the bicolored argument

Let us present a brief argument of why enumerating the minimal dominating sets in a bicolored graph  $G(A)$  is DOM-ENUM-hard if  $A$  can contain an arbitrarily large clique and no restriction is put on the structure of  $G - A$  nor its interactions with  $A$ . In other words, we argue that DOM-ENUM can be reduced to the problem of enumerating the minimal dominating sets in a bicolored graph  $G(A)$  where  $A$  is a clique.

Because of Theorem 2.1, we know that enumerating minimal dominating sets of a bipartite graph  $G$  is DOM-ENUM-hard. However, note that free to disregard the minimal dominating sets consisting of exactly one vertex in each clique of the partition, every minimal dominating set is included in one of the two cliques. Let  $A_1$  and  $A_2$  be the two sides of this partition. Observe that as both  $A_1$  and  $A_2$  induce cliques, they satisfy any property that does not limit the size of the largest clique. Combined with the fact that minimal dominating sets consisting of exactly one vertex in each side of the partition are easy to enumerate, we obtain the desired conclusion.

Note however that this obstacle was circumvented in Theorem 1.2 by keeping track of what the forbidden structures in  $G$  imply for the interactions between  $G - A$  and  $A$ . Unfortunately, the arguments were quite ad hoc in nature and it is unclear how far they can be generalized.

This obstacle was bypassed in a different way in Theorem 6.1, simply by first enumerating all the minimal dominating sets without a given structure, then using the fact that the structure appears in any remaining dominating set to guess where it does, and finally arguing that the vertices that remain to be dominated cannot induce an arbitrarily large clique. We now show that this technique is in fact very limited.

## 7.3 Limitations of enumerating all minimal dominating sets with a certain structure

We present now a brief argument on why enumerating all  $H$ -free minimal dominating sets in a graph is DOM-ENUM-hard unless  $H$  is a clique of size at most 2. Here, a minimal dominating set  $D$  is  $H$ -free if  $G[D]$  does not contain  $H$  as an induced subgraph.

The case when  $H$  is not a clique is directly implied by the argument in Section 7.2. We now focus on the case when  $H$  is a clique on at least 3 vertices; it suffices to handle

the case when  $H$  is a triangle. In other words, we argue that DOM-ENUM can be reduced to the question of enumerating all triangle-free minimal dominating sets.

Consider a graph  $G$ . We build an auxiliary graph  $G'$  by creating two copies  $A$  and  $B$  of  $V(G)$ , creating a vertex  $u$ , and setting  $V(G') = A \cup B \cup \{u\}$ . We set  $A$  to be an independent set,  $B$  to be a clique, and the vertex  $u$  to be adjacent to all of  $A$  and none of  $B$ . We set the edges between  $A$  and  $B$  as follows: a vertex in  $A$  and a vertex in  $B$  are adjacent if and only if the vertices of  $G$  they originate from are the same or are adjacent.

Let us consider what the structure of a minimal dominating set  $D$  of  $G'$  can be, and how easy it is to generate all minimal dominating sets of a given type. We consider three cases.

1.  $u \notin D$ . We generate all minimal dominating sets of the split graph  $G'[A \cup B]$ : this can be done in output-polynomial time according to Proposition 4.1. For each such minimal dominating set, either the intersection with  $A$  is non-empty and it is a minimal dominating set of  $G'$ , or it is empty and we can generate in polynomial-time all additions of a vertex of  $A$  that would result in a minimal dominating set of  $G'$ , if any. Since the number of minimal dominating sets of  $G'[A \cup B]$  with empty intersection with  $A$  is polynomially bounded by the number of those with non-empty intersection (see Lemma 4.3, Inequality (3)), we can generate all minimal dominating sets of  $G'$  not containing  $u$  in output-polynomial time.
2.  $u \in D$  and  $D \cap B \neq \emptyset$ . Then  $|D \cap B| = 1$ , and for any  $v \in B$ , the set  $\{u, v\}$  is a minimal dominating set of  $G'$ .
3.  $u \in D$  and  $D \cap B = \emptyset$ . All these minimal dominating sets are triangle-free. We observe that there is a bijection between the minimal dominating sets of this type and the minimal dominating sets of  $G$ .

Summarizing, the first two types of minimal dominating sets are easy to generate in output-polynomial time. We note that, free again to disregard minimal dominating sets that are easy to generate, enumerating all triangle-free minimal dominating sets of  $G'$  boils down to enumerating all minimal dominating sets of  $G'$  that are included in  $A \cup \{u\}$  and contain  $u$ . This is equivalent to enumerating all minimal dominating sets of  $G$ , hence the conclusion.

Note, however, that there is still hope for this technique when we assume some structure on the whole graph.

## 8 Perspectives for further research

In this paper, we investigated the enumeration of minimal dominating sets in graph classes forbidding an induced subgraph  $H$ . We gave algorithms that run in output-polynomial time and polynomial space when  $H$  is a clique, or more generally when  $H = K_t + K_2$ , and when  $H$  is the paw or the diamond. We now discuss possible directions for future research. For simplicity, let us here denote by  $\text{DOM-ENUM}(H)$  the problem DOM-ENUM restricted to  $H$ -free graphs.

The most natural continuation of our work is to seek output-polynomial time algorithms for  $\text{DOM-ENUM}(H)$  for other choices of the graph  $H$ . We discuss a possible classification of the graphs  $H$  depending on whether  $\text{DOM-ENUM}(H)$  admits an output-polynomial time algorithm, is DOM-ENUM-hard, or is not known to belong to one of these

two cases. We stress that the first two cases may not be disjoint as it is currently open whether DOM-ENUM admits an output-polynomial time algorithm in general. However, in the current state of the art, such a classification will highlight specific graph classes where the problem could be attacked more easily than in the general case.

Because of Theorem 2.1, if  $H$  is such that co-bipartite graphs form a subclass of  $H$ -free graphs then DOM-ENUM( $H$ ) is DOM-ENUM-hard. This includes the cases  $H = C_t$  or  $H = P_t$  with  $t \geq 5$ . This is also true for any graph  $H$  that has an independent set of size at least three, in particular all graphs  $H$  that have at least three connected components and graphs with two connected components where one component has one non-edge. Therefore, all the graphs  $H$  with more than one connected component for which DOM-ENUM( $H$ ) is not known to be DOM-ENUM-hard are of the form  $H = K_p + K_q$  (where by  $+$  we denote the disjoint union), for integers  $p, q \geq 1$ . We gave an output-polynomial time algorithm for the case where  $p \leq 2$  or  $q \leq 2$  in Theorem 6.1 and leave open the existence of such algorithms for  $p, q \geq 3$ .

Let us now focus on connected choices of  $H$ . Besides the case where  $H$  is a clique, which we addressed with Theorem 5.4, we settled the case where  $H = K_t - e$  for  $t = 4$  (Theorem 6.7). For  $t \in \{2, 3\}$ , DOM-ENUM( $H$ ) is output-polynomial time solvable since  $(K_t - e)$ -free graphs then are, respectively, cliques and disjoint unions of cliques. To the best of our knowledge, it is currently unknown whether DOM-ENUM( $K_t - e$ ) for  $t \geq 5$  is DOM-ENUM-hard and whether it is output-polynomial time solvable. We also considered graphs  $H$  of the form  $(K_t - \{uv, vw\})$  for  $t \geq 3$ , i.e., graphs obtained from a clique on  $t$  vertices by removing two incident edges. When  $t = 3$ ,  $(K_t - \{uv, vw\})$ -free graphs are exactly the complete multipartite graphs, for which an output-polynomial time algorithm can be obtained as in the proof of Lemma 6.10. We dealt with the case  $t = 4$  in Theorem 6.11 and leave open the cases of larger  $t$ .

Regarding the exclusion of specific graphs, we note that the status of DOM-ENUM( $P_t$ ) is completely explored: either  $t \leq 4$  and an output-polynomial time algorithm is known, or  $t \geq 5$  and the problem is DOM-ENUM-hard, as noted above. Among graph classes defined by forbidding an induced cycle, we proved that DOM-ENUM( $C_3$ ) is output-polynomial time solvable by Theorem 4.4 and noted above that DOM-ENUM( $C_t$ ) is DOM-ENUM-hard for  $t \geq 5$ , so only DOM-ENUM( $C_4$ ) remains to be classified. The graph  $C_4$  is also the only graph on at most 4 vertices for which DOM-ENUM( $H$ ) has not been classified yet. Other graph classes that are closed by taking induced subgraphs and where no output-polynomial time algorithm for DOM-ENUM neither DOM-ENUM-hardness proof are known to include unit-disk graphs [KN16, GHK<sup>+</sup>16] and comparability graphs.

Another natural research direction is to optimize the running times of our algorithms or to prove that this is not possible. Theorem 7.1 suggests that no improvement of our results can be obtained using backtrack search. We leave as an open problem whether there are polynomial delay algorithms for DOM-ENUM in the cases that we considered.

Finally, we note that the algorithm of Theorem 5.4 has been implemented in python/SageMath [Ray19].

## Acknowledgements

The authors wish to thank Paul Ouvrard for extensive discussions on the topic of this paper. We gratefully acknowledge support from Nicolas Bonichon and the Simon family for the organization of the 3<sup>rd</sup> Pessac Graph Workshop, where part of this research was done. We also thank the organisers of the Dagstuhl Seminar 18421 on algorithmic enu-

meration [FGS19] where some ideas present in this paper have been discussed. Last but not least, we thank Peppie for her unwavering support during the work sessions.

## References

- [Akk73] Eralp Abdurrahim Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM Journal on Computing*, 2(1):1–6, 1973.
- [Bar93] John M. Barnard. Substructure searching methods: Old and new. *Journal of Chemical Information and Computer Sciences*, 33(4):532–538, 1993.
- [BDHR19] Marthe Bonamy, Oscar Defrain, Marc Heinrich, and Jean-Florent Raymond. Enumerating minimal dominating sets in triangle-free graphs. In *36th International Symposium on Theoretical Aspects of Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- [CHvHK13] Jean-François Couturier, Pinar Heggernes, Pim van’t Hof, and Dieter Kratsch. Minimal dominating sets in graph classes: combinatorial bounds and enumeration. *Theoretical Computer Science*, 487:82–94, 2013.
- [Cou09] Bruno Courcelle. Linear delay enumeration and monadic second-order logic. *Discrete Applied Mathematics*, 157(12):2675–2700, 2009.
- [Dam06] Peter Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006.
- [EG02] Thomas Eiter and Georg Gottlob. Hypergraph transversal computation and related problems in logic and ai. In *European Workshop on Logics in Artificial Intelligence*, pages 549–564. Springer, 2002.
- [EGM03] 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.
- [EMG08] Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.
- [FGPS08] Fedor V. Fomin, Fabrizio Grandoni, Artem V. Pyatkin, and Alexey A. Stepanov. Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. *ACM Transactions on Algorithms (TALG)*, 5(1):9, 2008.
- [FGS19] Henning Fernau, Petr A. Golovach, and Marie-France Sagot. Algorithmic Enumeration: Output-sensitive, Input-Sensitive, Parameterized, Approximative ([Dagstuhl Seminar 18421](#)). *Dagstuhl Reports*, 8(10):63–86, 2019.
- [FK96] Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, 1996.

- [FLM97] Komei Fukuda, Thomas M. Liebling, and François Margot. Analysis of back-track algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry*, 8(1):1–12, 1997.
- [GHK<sup>+</sup>16] Petr A. Golovach, Pinar Heggernes, Mamadou M. Kanté, Dieter Kratsch, and Yngve Villanger. Enumerating minimal dominating sets in chordal bipartite graphs. *Discrete Applied Mathematics*, 199:30–36, 2016.
- [GHK<sup>+</sup>18] Petr A. Golovach, Pinar Heggernes, Mamadou M. Kanté, Dieter Kratsch, Sigve H. Sæther, and Yngve Villanger. Output-polynomial enumeration on graphs of bounded (local) linear mim-width. *Algorithmica*, 80(2):714–741, 2018.
- [GHKV15] Petr A. Golovach, Pinar Heggernes, Dieter Kratsch, and Yngve Villanger. An incremental polynomial time algorithm to enumerate all minimal edge dominating sets. *Algorithmica*, 72(3):836–859, 2015.
- [GK07] Joshua A. Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.
- [GKLS19] Petr A. Golovach, Dieter Kratsch, Mathieu Liedloff, and Mohamed Yosri Sayadi. Enumeration and maximum number of minimal dominating sets for chordal graphs. *Theoretical Computer Science*, 783:41–52, 2019.
- [JYP88] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.
- [KLM<sup>+</sup>13] Mamadou M. Kanté, Vincent Limouzy, Arnaud Mary, Lhouari Nourine, and Takeaki Uno. On the enumeration and counting of minimal dominating sets in interval and permutation graphs. In *International Symposium on Algorithms and Computation*, pages 339–349. Springer, 2013.
- [KLM<sup>+</sup>15] Mamadou M. Kanté, Vincent Limouzy, Arnaud Mary, Lhouari Nourine, and Takeaki Uno. A polynomial delay algorithm for enumerating minimal dominating sets in chordal graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 138–153. Springer, 2015.
- [KLMN11] Mamadou M. Kanté, Vincent Limouzy, Arnaud Mary, and Lhouari Nourine. Enumeration of minimal dominating sets and variants. In *International Symposium on Fundamentals of Computation Theory*, pages 298–309. Springer, 2011.
- [KLMN12] Mamadou M. Kanté, Vincent Limouzy, Arnaud Mary, and Lhouari Nourine. On the neighbourhood helly of some graph classes and applications to the enumeration of minimal dominating sets. In *International Symposium on Algorithms and Computation*, pages 289–298. Springer, 2012.
- [KLMN14] Mamadou M. Kanté, Vincent Limouzy, Arnaud Mary, and Lhouari Nourine. On the enumeration of minimal dominating sets and related notions. *SIAM Journal on Discrete Mathematics*, 28(4):1916–1929, 2014.

- [KN16] Mamadou M. Kanté and Lhouari Nourine. Minimal dominating set enumeration. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 1287–1291. Springer US, Boston, MA, 2016.
- [Mar64] Mitchell P. Marcus. Derivation of maximal compatibles using boolean algebra. *IBM Journal of Research and Development*, 8(5):537–538, 1964.
- [Mar15] Andrea Marino. *Analysis and enumeration: algorithms for biological graphs*, volume 6. Springer, 2015.
- [PU59] Marvin C. Paull and Stephen H. Unger. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Transactions on Electronic Computers*, (3):356–367, 1959.
- [Ray19] Jean-Florent Raymond. `minimal_dominating_sets`, an implementation of Bonamy et al.’s algorithm for enumerating minimal dominating sets in  $K_t$ -free graphs. <https://git.sagemath.org/sage.git/commit?id=906cf147fe64ceed73d30fabf61155f65393bd67> (accessed 25-feb-2020), 2019. To be included in SageMath 9.1 <http://www.sagemath.org>.
- [RT75] Ronald C. Read and Robert E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975.
- [SM19] Yann Strozecki and Arnaud Mary. Efficient enumeration of solutions produced by closure operations. *Discrete Mathematics & Theoretical Computer Science*, 21, 2019.
- [Str19] Yann Strozecki. Enumeration complexity. *Bulletin of EATCS*, 1(129), 2019.
- [Tar73] Robert E. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM Journal on Computing*, 2(3):211–216, 1973.
- [TIAS77] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.
- [Tie70] James C. Tiernan. An efficient search algorithm to find the elementary circuits of a graph. *Communications of the ACM*, 13(12):722–726, 1970.
- [Was16] Kunihiro Wasa. Enumeration of enumeration algorithms. *Preprint arxiv:1605.05102*, 2016.
- [YYH05] Xifeng Yan, Philip S. Yu, and Jiawei Han. Substructure similarity search in graph databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 766–777. ACM, 2005.