Enumerating Minimal Dominating Sets in Triangle-free Graphs*

Marthe Bonamy[1], Oscar Defrain[2],
Marc Heinrich[3], and Jean-Florent Raymond[4]

[1]*CNRS, LaBRI, Université de Bordeaux, France*
[2]*LIMOS, Université Clermont Auvergne, France*
[3]*LIRIS, Université Claude-Bernard, Lyon, France*
[4]*LaS team, Technische Universität Berlin, Germany.*

October 2018

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 prove that this is the case in triangle-free graphs. This answers a question of Kanté et al. Additionally, we show that deciding if a set of vertices of a bipartite graph can be completed into a minimal dominating set is a NP-complete problem.

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 [Mar15a, Dam04], and cheminformatics [Bar93]. Enumeration algorithms for graph problems seem to have been first mentioned in the early 70's with the pioneer works of Tiernen [Tie70] and Tarjan [Tar73] on cycles in directed graphs and of Akkoyunlu [Akk73]. 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 [Mar15b] for a more in-depth introduction to enumeration algorithms and to [Was16] for a listing of enumeration algorithms and problems.

*The second author has been supported by the ANR project GraphEn ANR-15-CE40-0009. 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 algorithms whose running time is polynomially bounded by the size of both the input and output data, called *output-polynomial* algorithms.

Because 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 hypergraph problem Trans-Enum. In Trans-Enum, one is given a hypergraph $\mathcal{H}$ (i.e. a collection of sets, called *hyperedges*) and is asked to enumerate all the minimal *transversals* of $\mathcal{H}$ (i.e. the inclusion-minimal sets of elements 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[1] instance of Dom-Enum [KLMN14]. Currently, the best output-sensitive algorithm for Trans-Enum is due to Fredman and Khachiyan and runs in 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 will be used to obtain new results on the Trans-Enum problem. So far, output-polynomial algorithms have been obtained for Dom-Enum in several classes of graphs, including planar graphs and degenerate graphs [EGM03], classes of graphs of bounded tree-width, clique-width [Cou09], or mim-width [GHK+18], path graphs and line graphs [KLMN12], interval graphs and permutation graphs [KLM+13], split graphs [KLM+15], graphs of girth at least 7 [GHKV15], chordal graphs [KLM+15], and chordal bipartite graphs [GHK+16]. A succinct survey of results on Dom-Enum can be found in [KN14]. The authors of [KLM+15] state as an open problem the question to design an output-polynomial algorithm for bipartite graphs (the problem also appeared in [KN14, GHK+16]). We address this problem with the following result.

Theorem 1.1. *There is an output-polynomial time algorithm enumerating minimal dominating sets in triangle-free graphs.*

In particular, the result holds for enumerating minimal dominating sets in bipartite graphs.

Our algorithm decomposes the graph by iteratively removing closed neighborhoods in the fashion of [EGM03], then constructs partial minimal dominating sets by adding the

[1] The complement of a bipartite graph.

neighborhoods back one after the other. It relies on the crucial property that in triangle-free graphs, the generation of all potential extensions of a partial minimal dominating set to a new neighborhood is closely related to the enumeration of minimal dominating sets in split graphs, for which tools have already been developed [KLMN14]. We note that triangle-free graphs already received attention in the context of enumeration of other objects, for instance maximal independent sets [HT93, Bys04], using different techniques.

A natural technique to enumerate valid solutions to a given problem (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 (that we will denote by DCS) is NP-complete [KLMN11], even when restricted to split graphs [KLM+15]. We show that it remains NP-complete in bipartite graphs.

Theorem 1.2. DCS *restricted to bipartite graphs is NP-complete.*

In particular, DCS is NP-complete in triangle-free graphs. This suggests that the aforementioned technique is unlikely to be used to improve Theorem 1.1.

The paper is organized as follows. In Section 2 we give the necessary definitions. We prove Theorems 1.1 and 1.2 in Sections 3 and 4, respectively. We conclude with possible future research directions in Section 5.

2 Preliminaries

Graphs. All graphs in this paper are finite, undirected, simple, and loopless. If G is a graph, then $V(G)$ is its set of vertices and $E(G) \subseteq V(G)^2$ 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. The subgraph of G *induced* by $X \subseteq V(G)$, denoted by $G[X]$, is the graph $(X, E(G) \cap (X \times X))$; $G \setminus X$ is the graph $G[V(G) \setminus X]$.

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 G is a split graph with clique C and stable set S and $X \subseteq V(G)$, we use X_C and X_S as shorthands for $X \cap C$ and $X \cap S$, respectively. Graphs where every cycle is of length at least 4 are referred to as *triangle-free* graphs. If f is a function, we write $f(n) = \operatorname{poly} n$ when there is a constant $c \in \mathbb{N}$ such that $f(n) = O(n^c)$.

Neighbors and domination. Let G be a graph and $x \in V(G)$. We note $N(x)$ 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 set of *closed neighbors* defined by $N[x] = N(x) \cup \{x\}$. For a given $X \subseteq V(G)$, we respectively denote by $N[X]$ and $N(X)$ the sets defined by $\bigcup_{x \in X} N[x]$ and $N[X] \setminus X$. Let D be a set of vertices of G. We say that D is *dominating* a subset $S \subseteq V(G)$ if $S \subseteq N[D]$. It is *minimally dominating* 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 $y \notin N[S \setminus \{x\}]$. Intuitively, this means that y is not dominated by any other vertex of S. Note that x can be its own private neighbor. The set of private neighbors of $x \in S$ in G is denoted by $\mathrm{Priv}_G(S, x)$ 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 $x \in S$, $\mathrm{Priv}(S, x) \neq \emptyset$.

Enumeration. The aim of graph enumeration algorithms is to generate a set of objects $\mathcal{X}(G)$ related to a graph G. We say that an algorithm enumerating $\mathcal{X}(G)$ with input an n-vertex graph G is *output-polynomial* if its running time is polynomially bounded by the size of the input and output data, i.e. $n + |\mathcal{X}(G)|$. If an algorithm enumerates $\mathcal{X}(G)$ by spending 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 some problems have output-polynomial algorithms but no polynomial delay ones, unless P=NP [Str10]. When discussing the space used by an enumeration algorithm, we ignore the space where the solutions are output.

3 Minimal domination in triangle-free graphs

In this section, we give an output-polynomial time algorithm to enumerate minimal dominating sets in triangle-free graphs. The algorithm is inspired from the one of [EGM03] and constructs dominating sets one neighborhood at a time.

A *peeling* of a graph G is a sequence $(V_0, \ldots, V_p)$ such that $V_p = V(G)$, $V_0 = \emptyset$, and for every $i \in \{1, \ldots, p\}$,

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

for some $v_i \in V_i$. We call $(v_1, \ldots, v_p)$ the *vertex sequence* of the peeling; note that p is only known after peeling the whole graph.

In the following, we consider a triangle-free graph G and a fixed peeling $(V_0, \ldots, V_p)$ with vertex sequence $(v_1, \ldots, v_p)$. For every $i \in \{0, \ldots, p\}$, we denote by $\mathcal{D}(G, i)$ the set of minimal dominating sets of V_i in G. Recall that these sets may contain vertices of $G - V_i$, which is a crucial point. Then $\mathcal{D}(G, p) = \mathcal{D}(G)$.

Definition 3.1. *Let $i \in \{0, \ldots, p-1\}$ and $D \in \mathcal{D}(G, i+1)$. We denote by* $\mathsf{Parent}(D, i+1)$ *the pair (D^*, i) where D^* is obtained from D by successively removing the vertex x of smaller index in D satisfying* $\mathrm{Priv}(D, x) \cap V_i = \emptyset$*, until no such vertex exists.*

Clearly, there is a unique way to build $\mathsf{Parent}(D, i+1)$ given D and i. By construction, the obtained set D^* is a minimal dominating set of V_i.

Proposition 3.2. *Let $i \in \{0, \ldots, p-1\}$ and $D^* \in \mathcal{D}(G, i)$.*
If D^* dominates V_{i+1} then $D^* \in \mathcal{D}(G, i+1)$ and $\mathsf{Parent}(D^*, i+1) = (D^*, i)$*.*
Otherwise, $D^* \cup \{v_{i+1}\} \in \mathcal{D}(G, i+1)$ and $\mathsf{Parent}(D^* \cup \{v_{i+1}\}, i+1) = (D^*, i)$*.*

Proof. First note that since $D^* \in \mathcal{D}(G, i)$, $\mathrm{Priv}(D^*, x) \cap V_i \neq \emptyset$ for all $x \in D^*$. Hence $\mathsf{Parent}(D^*, i+1) = (D^*, i)$ whenever D^* dominates V_{i+1}. If D^* does not dominate V_{i+1} then $D = D^* \cup \{v_{i+1}\}$ does. Moreover, $\mathrm{Priv}(D, v_{i+1}) \cap V_{i+1} \neq \emptyset$. Since v_{i+1} is not connected to any vertex in V_i, it cannot steal any private neighbors to the elements of D^*. Hence $\mathrm{Priv}(D, x) \cap V_{i+1} \neq \emptyset$ for all $x \in D$ and $\mathsf{Parent}(D^* \cup \{v_{i+1}\}, i+1) = (D^*, i)$. $\square$

The Parent relation as introduced in Definition 3.1 defines a tree on vertex set

$$\bigcup_{i=0}^{p} \{(D, i) \mid i \in \{1, \ldots, p\},\ D \in \mathcal{D}(G, i)\},$$

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

Given a set $D^* \in \mathcal{D}(G, i)$, we now focus on the enumeration of every $D \in \mathcal{D}(G, i+1)$ such that $(D, i+1)$ has (D^*, i) for parent. From Proposition 3.2, we know that either $(D^*, i+1)$ or $(D^* \cup \{v_{i+1}\}, i+1)$ has (D^*, i) for parent. Consequently, we refer to $X = \emptyset$ and $X = \{v_{i+1}\}$ as the *trivial extensions* of (D^*, i), and focus on the non-trivial ones.

We call *candidate extension* of (D^*, i) any (inclusion-wise) minimal set $X \subseteq V(G)$ such that $D^* \cup X$ dominates V_{i+1} in G, avoiding the trivial cases where $X \in \{\emptyset, \{v_{i+1}\}\}$. Then, X is a candidate extension of (D^*, i) if and only if $X \notin \{\emptyset, \{v_{i+1}\}\}$, $V_{i+1} \subseteq N[D^* \cup X]$ and, for every $x \in X$, $\mathrm{Priv}(D^* \cup X, x) \cap V_{i+1} \neq \emptyset$. Note that possibly not all candidate extensions of (D^*, i) form with D^* a minimal dominating set of V_{i+1}. In fact, there is no guarantee that any candidate extension forms a minimal dominating set of V_{i+1}: it might be that (D^*, i) has a unique child, given by its trivial extension. We denote by $\mathcal{C}(D^*, i)$ the set of candidate extensions of (D^*, i). We point out that by the minimality assumption, the vertex v_{i+1} appears in no element of $\mathcal{C}(D^*, i)$.

Lemma 3.3. *Let $i \in \{0, \ldots, p-1\}$ and $D^* \in \mathcal{D}(G, i)$. Then $|\mathcal{C}(D^*, i)| \leq |\mathcal{D}(G)|$.*

Proof. We argue that for every $X \in \mathcal{C}(D^*, i)$ there is an element of $\mathcal{D}(G, i+1)$ whose intersection with $V(G) \setminus D^*$ is precisely X. This will prove $|\mathcal{C}(D^*, i)| \leq |\mathcal{D}(G, i+1)|$, hence $|\mathcal{C}(D^*, i)| \leq |\mathcal{D}(G)|$ as desired.

Let $X \in \mathcal{C}(D^*, i)$. We consider the set $X \cup D^*$, which dominates V_{i+1}. By definition of $\mathcal{C}(D^*, i)$, we have $\mathrm{Priv}(X \cup D^*, x) \cap V_{i+1} \neq \emptyset$ for every $x \in X$. Therefore, every subset of $X \cup D^*$ that dominates V_{i+1} contains X. Consider an inclusion-wise minimal subset D' of $X \cup D^*$ that dominates V_{i+1}. We have $X \subseteq D'$, hence the conclusion. $\square$

Lemma 3.3 above ensures that $\mathcal{C}(D^*, i)$ is bounded by $\mathcal{D}(G)$. Hence, it is reasonable to test each of the candidate extensions even though D^* might be the parent of only one set in $\mathcal{D}(G, i+1)$. It now suffices to explain how to enumerate $\mathcal{C}(D^*, i)$ to complete the algorithm (formally described in Theorem 3.10).

Let $i \in \{0, \ldots, p-1\}$ and $D^* \in \mathcal{D}(G, i)$. We define $S = N(v_{i+1}) \cap V_{i+1} \setminus N[D^*]$ and $C = N(S) \setminus \{v_{i+1}\}$. As G is triangle-free and S is included in the neighborhood of v_{i+1}, S is an independent set. Let $Z^i_{D^*}$ be the split graph obtained from $G[C \cup S]$ where C is completed into a clique; note that the independent set S is maximal in $Z^i_{D^*}$ since $C \subseteq N(S)$. Recall that for any $X \subseteq V(Z^i_{D^*})$, we defined $X_C = X \cap C$ and $X_S = X \cap S$. We set $\mathcal{D}_{S=\emptyset}(Z^i_{D^*}) = \{D \in \mathcal{D}(Z^i_{D^*}) \mid D_S = \emptyset\}$. The following result is implicit in [KLMN14].

Proposition 3.4. *Let H be a split graph with maximal stable set S and clique C. Let $X \subseteq V(H)$. Then, $X \in \mathcal{D}(H)$ if and only if $S \subseteq N[X]$ and $\mathrm{Priv}(X, x) \cap S \neq \emptyset$ for all $x \in X$.*

Proof. Let us assume that $\mathrm{Priv}(X, x) \cap S \neq \emptyset$ for all $x \in X$ and that X dominates S. Then either $X \cap C \neq \emptyset$ and X also dominates C. Or $X \cap C = \emptyset$; in this case $X = S$ because X dominates S. As $C \subseteq N(S)$, X also dominates C. The minimality of X follows from our first assumption. Hence $X \in \mathcal{D}(H)$.

Conversely, let $X \in \mathcal{D}(H)$. Clearly $N[X] \supseteq S$, so we suppose by contradiction that $\mathrm{Priv}(X, x) \cap S = \emptyset$ for some $x \in X$. By minimality of X, we have $\mathrm{Priv}(X, x) \neq \emptyset$, which implies $\mathrm{Priv}(X, x) \subseteq C$. Consequently, we must have $X \cap C = \{x\}$. As $C \subseteq N(S)$, there exists a vertex $y \in S \cap N(v)$. Since $y \notin \mathrm{Priv}(X, x)$ and $X \cap C = \{x\}$, we have $y \in X$. However, in this case $N[y] \subseteq N[x]$ and so $\mathrm{Priv}(X, y) = \emptyset$, which contradicts the minimality of X. $\square$

We now characterize $\mathcal{C}(D^*, i)$ depending on whether v_{i+1} has to be dominated by the extension or not. The condition $D^* \in \mathcal{D}(G, i) \setminus \mathcal{D}(G, i+1)$ in the statement below prevents (D^*, i) from having the trivial extension $\emptyset$ - in which case it is the only one.

Lemma 3.5. *Let $i \in \{0, \ldots, p-1\}$, $D^* \in \mathcal{D}(G, i) \setminus \mathcal{D}(G, i+1)$ and $Z = Z^i_{D^*}$. Then*

- *either $D^* \cap N(v_{i+1}) \neq \emptyset$ and $\mathcal{C}(D^*, i) = \mathcal{D}(Z)$,*
- *or $D^* \cap N(v_{i+1}) = \emptyset$ and*

$$\mathcal{C}(D^*, i) = (\mathcal{D}(Z) \setminus \mathcal{D}_{S=\emptyset}(Z)) \cup \left\{ Q \cup \{u\} \;\middle|\; \begin{array}{l} Q \in \mathcal{D}_{S=\emptyset}(Z), \\ u \in N(v_{i+1}), \text{ and} \\ \forall x \in Q,\ \mathrm{Priv}(Q \cup \{u\}, x) \cap V_{i+1} \neq \emptyset \end{array} \right\}.$$

Proof. Let us first consider the case $D^* \cap N(v_{i+1}) \neq \emptyset$. Let $X \in \mathcal{C}(D^*, i)$. Since v_{i+1} is dominated by any vertex of $D^* \cap N(v_{i+1})$, only the stable set S of Z is to be dominated by X. In other words X minimally dominates S: $S \subseteq N[X]$ and $\mathrm{Priv}(X, x) \cap S \neq \emptyset$ for all $x \in X$. By Proposition 3.4, $X \in \mathcal{D}(Z)$, which proves the inclusion $\mathcal{C}(D^*, i) \subseteq \mathcal{D}(Z)$. Conversely, let $X \in \mathcal{D}(Z)$. By Proposition 3.4, $S \subseteq N[X]$ and $\mathrm{Priv}(X, x) \cap S \neq \emptyset$ for all $x \in X$. Since v_{i+1} is already dominated by D^*, $X \in \mathcal{C}(D^*, i)$. Hence $\mathcal{C}(D^*, i) = \mathcal{D}(Z)$, as desired.

From now on and until the end of the proof we assume that $D^* \cap N(v_{i+1}) = \emptyset$. Let C denote the vertex set of the clique of Z. Let $X \in \mathcal{C}(D^*, i)$. We know that X must be a dominating set of Z. Indeed, by definition of $\mathcal{C}(D^*, i)$, X dominates S, and either $X \cap C \neq \emptyset$, in which case X also dominates C, or $X = S$ and X also dominates C since $C \subseteq N_Z(S)$. There are two cases to consider.

If X is a minimal dominating set of Z, then since X has to dominate v_{i+1}, we have $X \cap S \neq \emptyset$ and consequently $X \in \mathcal{D}(Z) \setminus \mathcal{D}_{S=\emptyset}(Z)$.

Otherwise, X is not a *minimal* dominating set of Z. This implies that it has a vertex u with no private neighbor in Z. By definition of $\mathcal{C}(D^*, i)$, this means that $\mathrm{Priv}(D^* \cup X, u) \cap V_{i+1} = \{v_{i+1}\}$. Therefore there is exactly one such vertex. Then, if we write $Q = X \setminus \{u\}$, Q is a minimal dominating set of Z. Since v_{i+1} is a private neighbor of u, we must have $Q \cap S = \emptyset$, and consequently $Q \in \mathcal{D}_{S=\emptyset}(Z)$. Finally, by definition of $C(D^*, i)$, for any $x \in Q \subset X$, we have $\mathrm{Priv}(X, x) \cap V_{i+1} \neq \emptyset$. This shows that we have

$$X \in \left\{ Q \cup \{u\} \;\middle|\; \begin{array}{l} Q \in \mathcal{D}_{S=\emptyset}(Z), \\ u \in N(v_{i+1}), \text{ and} \\ \forall x \in Q,\ \mathrm{Priv}(Q \cup \{u\}, x) \cap V_{i+1} \neq \emptyset \end{array} \right\}, \tag{1}$$

and proves the following inclusion:

$$\mathcal{C}(D^*, i) \subseteq (\mathcal{D}(Z) \setminus \mathcal{D}_{S=\emptyset}(Z)) \cup \left\{ Q \cup \{u\} \;\middle|\; \begin{array}{l} Q \in \mathcal{D}_{S=\emptyset}(Z), \\ u \in N(v_{i+1}), \text{ and} \\ \forall x \in Q,\ \mathrm{Priv}(Q \cup \{u\}, x) \cap V_{i+1} \neq \emptyset \end{array} \right\}.$$

To prove the reverse inclusion, we first consider $X \in \mathcal{D}(Z) \setminus \mathcal{D}_{S=\emptyset}(Z)$. By Proposition 3.4, $S \subseteq N[X]$ and $\mathrm{Priv}(X, x) \cap S \neq \emptyset$ for all $x \in X$. Since $X \cap S \neq \emptyset$, $S \cup \{v_{i+1}\} \subseteq N[X]$. Thus $X \in \mathcal{C}(D^*, i)$. Now we consider a set X of the form $Q \cup \{u\}$, for some $Q \in \mathcal{D}_{S=\emptyset}(Z)$ and $u \in N(v_{i+1})$ such that $\forall x \in Q$, $\mathrm{Priv}(Q \cup \{u\}, x) \cap V_{i+1} \neq \emptyset$. By Proposition 3.4, $\mathrm{Priv}_Z(Q, x) \cap S \neq \emptyset$ for all $x \in Q$. Since $\mathrm{Priv}(Q \cup \{u\}, x) \cap V_{i+1} \neq \emptyset$ for all $x \in Q$ and $v_{i+1} \in \mathrm{Priv}(X, u)$, $\mathrm{Priv}(X, x) \cap V_{i+1} \neq \emptyset$ for all $x \in X$. Since $S \cup \{v_{i+1}\} \subseteq N[X]$, $X \in \mathcal{C}(D^*, i)$. This proves the reverse inclusion and concludes the proof. □

In [KLMN14], authors give a polynomial delay algorithm to enumerate minimal dominating sets in split graphs.

Theorem 3.6 ([KLMN14])**.** *There is an algorithm that, given a split graph H with n vertices and m edges, outputs with $O(n + m)$ delay every minimal dominating set of H, using $O(n^2)$ space.*

The above algorithm relies on the observation that for every split graph H, the set $\mathcal{D}_C(H) = \{D_C \mid D \in \mathcal{D}(H)\}$ is in bijection with $\mathcal{D}(H)$ and it forms an independent set system. A family of sets $\mathcal{S}$ is an *independent set system* if $S \in \mathcal{S}$ implies that $S \setminus \{s\} \in \mathcal{S}$ for all $s \in S$. We show that there is a polynomial delay algorithm to enumerate $\mathcal{C}(D^*, i)$ given $i \in \{1, \ldots, p-1\}$ and $D^* \in \mathcal{D}(G, i)$ using the same observations.

Proposition 3.7 ([KLMN14]). *Let H be a split graph with maximal stable set S and clique C and let D be a minimal dominating set of H. Then $D_S = S \setminus N(D_C)$.*

Proposition 3.8 ([KLMN14]). *Let H be a split graph with maximal stable S and clique C. Then:*

1. $\mathcal{D}_C(H) = \{A \subseteq C \mid \forall x \in A,\ \mathrm{Priv}(A, x) \neq \emptyset\}$,
2. *$\mathcal{D}_C(H)$ and $\mathcal{D}(H)$ are in bijection,*
3. *$\mathcal{D}_C(H)$ is an independent set system.*

Lemma 3.9. *There is an algorithm that, given $i \in \{0, \ldots, p-1\}$ and $D^* \in \mathcal{D}(G, i)$, enumerates $\mathcal{C}(D^*, i)$ in output-polynomial time $O(\mathrm{poly}(n) \cdot |\mathcal{C}(D^*, i)|)$ and using at most $\mathrm{poly}(|V(G)|)$ space.*

Proof. Lemma 3.5 allows us to consider two cases depending on whether v_{i+1} has a neighbor in D^* or not. Let $Z = Z^i_{D^*}$. As usual we denote by S and C the maximal stable set and the clique of Z, respectively.

If $D^* \cap N(v_{i+1}) \neq \emptyset$, then by Lemma 3.5 $\mathcal{C}(D^*, i) = \mathcal{D}(Z)$, and we can enumerate the elements of $\mathcal{C}(D^*, i)$ with polynomial delay using the algorithm of Theorem 3.6 on $\mathcal{D}(Z)$.

In the case where $D^* \cap N(v_{i+1}) = \emptyset$, we start enumerating $\mathcal{D}_C(Z)$. This can be done with polynomial delay and space as in the proof of Theorem 3.6, using the fact that $\mathcal{D}_C(Z)$ is an independent set system and that testing if an arbitrary set A belongs to $\mathcal{D}_C(Z)$ can be done in polynomial time using Lemma 3.8. That is, we construct elements of $\mathcal{D}_C(Z)$ from the empty set to every inclusion-wise maximal $A \in \mathcal{D}_C(Z)$. Repetitions are avoided using a linear ordering on vertices of C; see [KLMN14] for details. Then, for every element $A \in \mathcal{D}_C(Z)$ output by the above algorithm, we check in polynomial time if it dominates Z. If it does not, then we extend A into its unique corresponding minimal dominating set $D \in \mathcal{D}(Z)$ such that $D \cap C = A$ (i.e. $D = A \cup S \setminus N(A)$), and output D. Otherwise, for every $u \in N(v_{i+1})$ such that for all $x \in A$, $\mathrm{Priv}(A \cup \{u\}, x) \cap V_{i+1} \neq \emptyset$ (which can be tested in time polynomial in the order of Z), we output $A \cup \{u\}$. Lemma 3.5 guarantees that the above algorithm indeed outputs $\mathcal{C}(D^*, i)$.

Note that the only elements $D \in \mathcal{D}(Z)$ which do not lead to an element of $\mathcal{C}(D^*, i)$ are the $D \in \mathcal{D}_{S=\emptyset}(Z)$ for which no vertex $u \in N(v_{i+1})$ satisfies the desired conditions. However, we will show that $|\mathcal{D}_{S=\emptyset}(Z)| \leq n|\mathcal{D}(Z) \setminus \mathcal{D}_{S=\emptyset}(Z)|$. Indeed, consider the map f that, given $D \in \mathcal{D}_{S=\emptyset}(Z)$ removes one arbitrary vertex from D, and completes the dominating set by adding the vertices in the independent set which are no longer dominated. Then, f maps elements of $\mathcal{D}_{S=\emptyset}(Z)$, to the set $\mathcal{D}(Z) \setminus \mathcal{D}_{S=\emptyset}(Z)$. Moreover, every element in this second set is the image of at most $|C| \leq n$ elements by f. This implies the desired bound.

Consequently, this means that while enumerating $\mathcal{D}(Z)$, we might throw out a fraction at most $\frac{n}{n+1}$ of all the solutions we found which do not lead to elements in $\mathcal{C}(D^*, i)$. This shows that the algorithm has output-polynomial time. $\square$

We are now ready to prove Theorem 1.1, that we restate here in a more accurate form.

Theorem 3.10. *There is an algorithm that, given a triangle-free graph G on n vertices, outputs $\mathcal{D}(G)$ in total time $\text{poly}(n) \cdot |\mathcal{D}(G)|^2$ and using at most $\text{poly}\, n$ space.*

Proof. We first arbitrarily choose a peeling $(V_0, \ldots, V_p)$ of our input graph G with vertex sequence $(v_1, \ldots, v_p)$. This takes time $\text{poly}\, n$.

Recall that the Parent relation defines a tree T on vertex set

$$\bigcup_{i=0}^{p} \{(D, i) \mid i \in \{1, \ldots, p\}\,,\ D \in \mathcal{D}(G, i)\},$$

with leaves $\{(D, p) \mid D \in \mathcal{D}(G)\}$ and root $(\emptyset, 0)$. Let us describe how to enumerate the children in T of (D^*, i) for every given vertex $D^* \in \mathcal{D}(G, i)$. If D^* dominates V_{i+1}, then $(D^*, i+1)$ is the only pair whose parent is (D^*, i). Otherwise, we proceed as follows:

1. output the trivial child $D^* \cup \{v_{i+1}\}$;
2. start (or resume, if it had already been started) the algorithm of Lemma 3.9 and pause it after one element X of $C(D^*, i)$ has been output;
3. if $D^* \cup X$ is not a minimal dominating set of V_{i+1} in G, or if it is but $\mathsf{Parent}(D^* \cup X, i+1) \neq (D^*, i)$, discard X and loop to (2);
4. output $D^* \cup X$ and loop to (2).

The algorithm terminates when the algorithm of Lemma 3.9 in step (2) completes the enumeration of $C(D^*, i)$. The correctness of the algorithm is a consequence of the following inclusions:

$$\begin{aligned}\{D \in \mathcal{D}(G, i+1) \mid \mathsf{Parent}(D, i+1) = (D^*, i)\} \subseteq &\{D \in \mathcal{D}(G, i+1) \mid D^* \subseteq D\}\\ \subseteq &\{D^* \cup X \mid X \in \mathcal{C}(D^*, i)\}\\ &\cup \{D^* \cup \{v_{i+1}\}\}\\ &\cup \{D^*\}\end{aligned}$$

Notice that it uses at most $\text{poly}\, n$ space, since we only store the data of the algorithm of Lemma 3.9, of size at most $\text{poly}\, n$, and the data to perform step (3), which is clearly also polynomial in n.

In order to enumerate $\mathcal{D}(G)$, i.e. the set of leaves of T, we perform a DFS and output each visited leaf. For each vertex of T, enumerating its children can be done in at most $\text{poly}(n) \cdot |\mathcal{D}(G)|$ steps with the above algorithm, according to Lemmas 3.3 and 3.9. Besides, the number of vertices of T at distance i from the root is at most its number of leaves, hence T has at most $O(n \cdot |\mathcal{D}(G)|)$ vertices. Therefore we can enumerate $\mathcal{D}(G)$ in $\text{poly}(n) \cdot |\mathcal{D}(G)|^2$ steps. Regarding the space, we observe that whenever we visit a vertex, 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 the algorithm above (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) algorithms enumerating the children of the ancestors of (D, i) and the data of the algorithm enumerating the children of D, i.e. $i \cdot \text{poly}\, n$ space. Therefore the described algorithm uses polynomial space, as claimed. $\square$

4 The extension problem is hard in bipartite graphs

We recall that Dcs denotes the problem of deciding, given a graph G and a set $A \subseteq V(G)$, whether there exists a minimal dominating set D of G such that $A \subseteq D$. This problem is known to be NP-complete for general graphs [KLMN11]. It has later been proved that the variant where we search for a minimal dominating set containing A, and avoiding a given vertex set B remains intractable even on split graphs [KLM+15]. We show that Dcs is still hard for bipartite graphs and thus triangle-free graphs. As a consequence, one cannot expect to improve Theorem 1.1 by testing if subsets of $V(G)$ can be extended into minimal dominating sets of G. The following is a restatement of Theorem 1.2.

Theorem 4.1. 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 reduction from SAT.

Given an instance $\mathcal{I}$ of SAT with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, 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 that satisfies all the clauses. The graph G has vertex partition (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 neg_{x_i} per variable x_i, and two special vertices v and z. For every $i \in \{1, \ldots, n\}$ we make neg_{x_i} adjacent to the two literals x_i and $\neg x_i$ and for every $j \in \{1, \ldots, m\}$ 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 $A = \{neg_{x_1}, \ldots, neg_{x_n}, v, w\}$. Clearly this graph can be constructed in polynomial time from $\mathcal{I}$. The construction is illustrated in Figure 1.

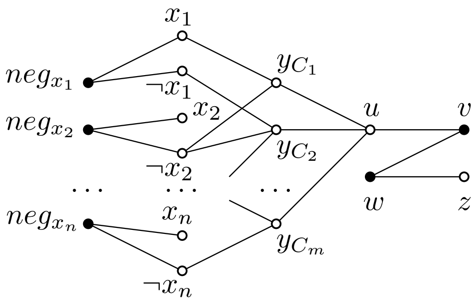

Figure 1: A bipartite graph G and a set $A \subseteq V(G)$ constructed from an instance of SAT with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. Black vertices constitute the set A. Then A can be extended into a minimal dominating set D of G if and only if there is a truth assignment of the variable satisfying all the clauses.

Let us show that A can be extended into a minimal dominating set of G if and only if $\mathcal{I}$ has a truth assignment that satisfies all the clauses. The proof is split into two claims. A *partial assignment* of $\mathcal{I}$ is a truth assignment of a subset of the variables $x_1, \ldots, x_n$. 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 no proper subset of the assigned variables admits such a partial assignment.

Claim 4.2. Let $S \subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ be a set containing at most one literal for each variable. Then S minimally dominates $\{y_{C_1}, \ldots, y_{C_m}\}$ if and only if its elements form a minimal assignment of $\mathcal{I}$.

Proof of Claim 4.2. Let S be as above and let $j \in \{1, \ldots, m\}$. 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 a partial assignment of the variables of $\mathcal{I}$ satisfying all its clauses is given by the literals present in S. Moreover, x has a private neighbor $y_{C_{j'}}$, by minimality of S. The assignment given by S is hence minimal: not specifying the value of the variable of x would leave the clause $C_{j'}$ unsatisfied. ⌟

Claim 4.3. If D is a minimal dominating set of G containing A, then $D \setminus A \subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ and it contains at most one literal for each variable.

Proof of Claim 4.3. Notice that $\mathrm{Priv}(A, v) = \{u\}$. If y_{C_j} belongs to D for some $j \in \{1, \ldots, m\}$, then $\mathrm{Priv}(D, v) = \emptyset$, a contradiction to the minimality of D. For similar reasons $u, z \notin D$. Hence $D \cap \{u, z, y_{C_1}, \ldots, y_{C_m}\} = \emptyset$. Besides, for every $i \in \{1, \ldots, m\}$, D contains at most one of x_i and $\neg x_i$, as otherwise $\mathrm{Priv}(D, neg_{x_i})$ would be empty, again contradicting the minimality of D. This proves the claim. ⌟

If A can be extended into a minimal dominating set D of G, then by combining the two claims above, we deduce that $\mathcal{I}$ has truth assignment that satisfies all clauses. Conversely, if $\mathcal{I}$ has such a truth assignment, then there is a set S as in the statement of Claim 4.2. 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 variables. Besides, each of $neg_{x_1}, \dots, neg_{x_n}$ 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 $\mathcal{I}$ of SAT, we constructed in polynomial time an instance (G, A) of Dcs that is equivalent to $\mathcal{I}$. This proves that Dcs is NP-hard. □

5 Conclusion

In this paper, we proved that the set of minimal dominating sets of a graph can be enumerated in output-polynomial time in triangle-free graphs, and hence in bipartite graphs. It remains open whether a polynomial delay algorithm exists for these classes.

The most general open problem on the topic discussed in this paper is whether the minimal dominating sets of a co-bipartite graph can be enumerated in output-polynomial time. Indeed, as noted in the introduction this would imply that such an algorithm also exists for the general case. Other classes where no output-polynomial time algorithms are known include unit disk graphs and graphs of bounded expansion, according to [KN14, GHK+16].

Acknowledgements

The authors wish to thank Paul Ouvrard for extensive discussions on the topic of this paper. We also gratefully acknowledge support from Nicolas Bonichon and the Simon family for the organization of the 3rd Pessac Graph Workshop, where this research was done. 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.

[Bys04] Jesper Makholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32(6):547–556, 2004.

[Cou09] Bruno Courcelle. Linear delay enumeration and monadic second-order logic. *Discrete Applied Mathematics*, 157(12):2675–2700, 2009.

[Dam04] Peter Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. In Rod Downey, Michael Fellows, and Frank Dehne, editors, *Parameterized and Exact Computation*, pages 1–12, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[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. arxiv:cs/0204009.

[EMG08] Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.

[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.

[GHK+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. Special Issue: Sixth Workshop on Graph Classes, Optimization, and Width Parameters 2013.

[GHK+18] Petr A. Golovach, Pinar Heggernes, Mamadou Moustapha 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, Feb 2018. arxiv:1509.03753.

[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, Jul 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.

[HT93] Mihály Hujtera and Zsolt Tuza. The number of maximal independent sets in triangle-free graphs. *SIAM Journal on Discrete Mathematics*, 6(2):284–288, 1993.

[KLM+13] Mamadou Moustapha 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+15] Mamadou Moustapha 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. arxiv:1407.2036.

[KLMN11] Mamadou Moustapha 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. arxiv:1407.2053.

[KLMN12] Mamadou Moustapha 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 Moustapha 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. arxiv:1407.2053.

[KN14] Mamadou Moustapha Kanté and Lhouari Nourine. Minimal dominating set enumeration. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 1–5. Springer US, Boston, MA, 2014.

[Mar64] M. P. Marcus. Derivation of maximal compatibles using boolean algebra. *IBM Journal of Research and Development*, 8(5):537–538, Nov 1964.

[Mar15a] Andrea Marino. An application: Biological graph analysis. In *Analysis and Enumeration: Algorithms for Biological Graphs*, pages 37–44. Atlantis Press, Paris, 2015.

[Mar15b] Andrea Marino. Enumeration algorithms. In *Analysis and Enumeration: Algorithms for Biological Graphs*, pages 13–35. Atlantis Press, Paris, 2015.

[PU59] M. C. Paull and S. H. Unger. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Transactions on Electronic Computers*, EC-8(3):356–367, Sept 1959.

[Str10] Yann Strozecki. *Enumeration complexity and matroid decomposition*. PhD thesis, Paris 7, 2010.

[Tar73] Robert Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM Journal on Computing*, 2(3):211–216, 1973.

[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. See also https://kunihirowasa.github.io/enum/index (accessed on September 2018).

[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.