Minimal Dominating Sets in Graph Classes: Combinatorial Bounds and Enumeration*

Jean-François Couturier[1], Pinar Heggernes[2],
Pim van 't Hof[2], and Dieter Kratsch[1]

[1] LITA, Université de Lorraine, 57045 Metz Cedex 01, France.
{couturier,kratsch}@univ-metz.fr
[2] Department of Informatics, University of Bergen, N-5020 Bergen, Norway.
{pinar.heggernes,pim.vanthof}@ii.uib.no

Abstract. The number of minimal dominating sets that a graph on n vertices can have is known to be at most 1.7159^n. This upper bound might not be tight, since no examples of graphs with 1.5705^n or more minimal dominating sets are known. For several classes of graphs, we substantially improve the upper bound on the number of minimal dominating sets. At the same time, we give algorithms for enumerating all minimal dominating sets, where the running time of each algorithm is within a polynomial factor of the proved upper bound for the graph class in question. In several cases, we provide examples of graphs containing the maximum possible number of minimal dominating sets for graphs in that class, thereby showing the corresponding upper bounds to be tight.

1 Introduction

Combinatorial questions of the type *"What is the maximum number of vertex subsets satisfying a given property in a graph?"* have found interest and applications in computer science, especially in exact exponential algorithms [9]. The question has been studied recently, both on general graphs and on some graph classes, for minimal feedback vertex sets, minimal subset feedback vertex sets, minimal separators, and potential maximal cliques [4, 6, 8, 10, 12, 13]. A famous classical example is the highly cited theorem of Moon and Moser [27], which states that the maximum number of maximal cliques and maximal independent sets, respectively, in any graph on n vertices is $3^{n/3}$. Although the original proof of the upper bound in [27] is by induction, it is not hard to transform it into a branching algorithm enumerating all maximal independent sets of a graph in time $O^*(3^{n/3})$, where the O^*-notation suppresses polynomial factors. These results were used by Lawler [23] to give an algorithm for graph coloring, which was the fastest algorithm for this purpose for over two decades. A faster algorithm for graph coloring was obtained by Eppstein [5] by improving the upper bound on the number of maximal independent sets of small size.

* An extended abstract of this paper was presented at SOFSEM 2012 [3]. This work is supported by the Research Council of Norway and the French National Research Agency.

The number of papers on domination in graphs is in the thousands, and several well known surveys and books are dedicated to the topic (see, e.g., [16]). It is surprising that a first non-trivial answer to the Moon and Moser type question *"What is the maximum number of minimal dominating sets in a graph?"* was established only recently. Fomin, Grandoni, Pyatkin and Stepanov [7] gave an $O(1.7159^n)$ time algorithm for enumerating all minimal dominating sets in an n-vertex graph, thereby showing that the maximum number of minimal dominating sets in such a graph is at most 1.7159^n. They used this result to derive an $O(2.8718^n)$ algorithm for the Domatic Number problem [7]. Although examples of graphs with 1.5704^n minimal dominating sets have been identified [7] (see Fig. 1), it is not known whether graphs with 1.5705^n or more minimal dominating sets exist.

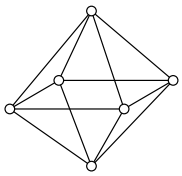

Fig. 1. The graph G^*, which has 6 vertices and 15 minimal dominating sets. The graph G_n^* on $n = 6k$ vertices, consisting of k disjoint copies of G^*, has $15^{n/6} \approx 1.5704^n$ minimal dominating sets.

Our interest in this combinatorial question was triggered by the large gap between the best known lower and upper bounds. Our achievements here are twofold: we narrow this gap on a variety of graph classes, and we give enumeration algorithms whose running times match the upper bounds that we prove. Our upper bound proofs heavily rely on structural graph properties. In a very recent paper, Krzywkowski [21] obtained similar results for the class of trees. Before we list the results achieved in this paper in more detail, let us mention that enumeration of minimal dominating sets has been studied on graph classes from a different perspective in several recent works [14, 18, 19]. These concentrate on enumerating the minimal dominating sets of a graph in time polynomial in the number of minimal dominating sets of the input graph.

In Sections 3, 4, and 5 of this paper, we present branching algorithms for enumerating all minimal dominating sets in chordal graphs, split graphs, and proper interval graphs. The branching rules of these algorithms immediately yield upper bounds on the maximum number of minimal dominating sets for graphs in these classes that are significantly lower than the best known upper bound for general graphs. In Section 6, we use combinatorial arguments to obtain upper bounds for cographs, trivially perfect graphs, threshold graphs, and chain graphs. These arguments can trivially be turned into algorithms for enumerating all minimal dominating sets, where the running time of each algorithm is within a polynomial factor of the proved upper bound for the corresponding graph class.

For each of these four classes, we provide examples of graphs that contain the maximum possible number of minimal dominating sets, thereby showing that the corresponding upper bounds are tight. Our findings are summarized in rows 2-8 of Table 1. The inclusion relationships between the graph classes studied in this paper are illustrated in Fig. 2.

Graph Class	Lower Bound	Upper Bound
general [7]	$15^{n/6}$	1.7159^n
chordal	$3^{n/3}$	1.6181^n
split	$3^{n/3}$	1.4656^n
proper interval	$3^{n/3}$	1.4656^n
cograph	$15^{n/6}$	$15^{n/6}$
trivially perfect	$3^{n/3}$	$3^{n/3}$
threshold	$\omega(G)$	$\omega(G)$
chain	$\lfloor n/2 \rfloor + m$	$\lfloor n/2 \rfloor + m$

Table 1. Lower and upper bounds on the maximum number of minimal dominating sets. Note that $15^{n/6} \approx 1.5704^n$ and $3^{n/3} \approx 1.4422^n$.

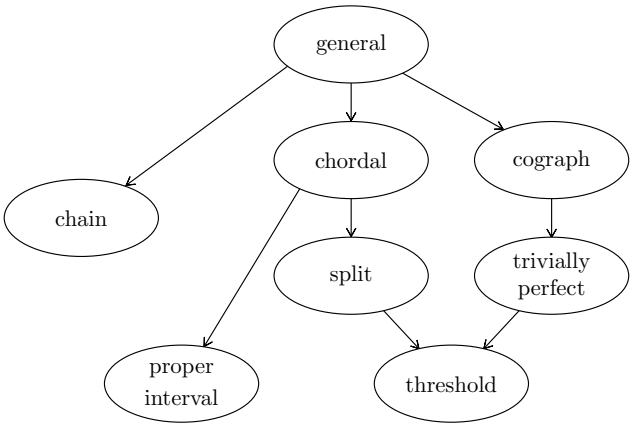

Fig. 2. The graph classes studied in this paper, where $\rightarrow$ represents the $\supset$ relation.

2 Preliminaries

We work with simple undirected graphs. We denote such a graph by $G = (V, E)$, where V is the set of vertices and E is the set of edges of G. We adhere to the

convention that $n = |V|$ and $m = |E|$. When the vertex set and the edge set of G are not specified, we use $V(G)$ and $E(G)$ to denote these, respectively. The set of *neighbors* of a vertex $v \in V$ is the set of vertices adjacent to v, and is denoted by $N_G(v)$. The *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. For a set $S \subseteq V$, we define analogously $N_G(S) = \bigcup_{v \in S} N_G(v) \setminus S$ and $N_G[S] = N_G(S) \cup S$. We will omit the subscript G when there is no ambiguity. A vertex v is *universal* if $N[v] = V$ and *isolated* if $N(v) = \emptyset$. The subgraph of G *induced* by S is denoted by $G[S]$. For ease of notation, we use $G - v$ to denote the graph $G[V \setminus \{v\}]$, and $G - S$ to denote the graph $G[V \setminus S]$. A graph is *connected* if there is a path between every pair of its vertices. A maximal connected subgraph of G is called a *connected component* of G. A set $S \subseteq V$ is called an *independent set* if $uv \notin E$ for every pair of vertices $u, v \in S$, and S is called a *clique* if $uv \in E$ for every pair of vertices $u, v \in S$. A clique is *maximal* if no proper superset of it is a clique, and *maximum* if no clique exists with strictly larger size. Maximal and maximum independent sets are defined analogously. We use $\omega(G)$ to denote the size of a maximum clique in G.

A vertex set $S \subseteq V$ is a *dominating set* of G if $N[S] = V$. Every vertex v of a dominating set *dominates* the vertices in $N[v]$. A dominating set S is a *minimal dominating set* if no proper subset of S is a dominating set. It is an easy observation that, if S is a minimal dominating set, then for every vertex $v \in S$, there is a vertex $x \in N[v]$ which is dominated only by v. We will call such a vertex x a *private neighbor* of v, since x is not adjacent to any vertex in $S \setminus \{v\}$. Note that a vertex in S might be its own private neighbor. The number of minimal dominating sets in a graph G is denoted by $\mu(G)$. The following observation follows from the fact that every minimal dominating set of G is the union of a minimal dominating set of each connected component of G.

Observation 1 *Let G be a graph with connected components G_1, G_2, ... , G_t. Then $\mu(G) = \prod_{i=1}^{t} \mu(G_i)$.*

Each of the graph classes that we study will be defined in the section containing the results on that class. All of the graph classes mentioned in this paper can be recognized in linear time, and they are closed under taking induced subgraphs [1, 15]. Here we define two graph families that will be useful for proving lower bounds on the maximum number of minimal dominating sets. We write H_n to denote the graph on $n = 3k$ vertices which is the disjoint union of k triangles. We write S_n to denote the graph on $n = 3k$ vertices which consists of a clique C of size $2k$ and an independent set I of size k, such that each vertex of I has exactly two neighbors in C, and no two vertices in I have a common neighbor. It can be verified easily that $\mu(H_n) = \mu(S_n) = 3^{n/3} \approx 1.4422^n$. The graphs H_3 and S_2 are depicted in Fig. 3.

2.1 Preliminaries on Branching

In Sections 3, 4, and 5, we present branching algorithms for enumerating all minimal dominating sets in chordal graphs, split graphs, and proper interval

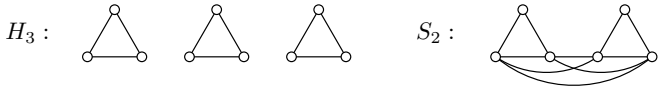

Fig. 3. The graphs H_3 and S_2.

graphs. We refer to the monograph by Fomin and Kratsch [9] for an introduction on branching algorithms.

For each of the three branching algorithms, every recursive call has input (G', D), where G' is an induced subgraph of the input graph G, and D is a subset of $V(G) \setminus V(G')$. At each step, vertices of G' are either *added* to D, or *discarded*, which means that they are not added to D and will not be added to D at a later step. At any time during the execution of the algorithms, we say that a vertex v of G is *dominated* if D contains a vertex of $N_G[v]$, and *undominated* otherwise. The *measure* of an instance is defined either as the number or the total weight of the vertices in that instance. The algorithms either delete vertices from G' or reduce the weights of vertices at every step, resulting in a strict decrease of the measure. Each algorithm generates a collection $\mathcal{D}$ of subsets of $V(G)$, making sure that every minimal dominating set of G appears in $\mathcal{D}$. Hence, by checking for each element of $\mathcal{D}$ whether it is a minimal dominating set of G, and outputting it if and only if this is the case, the algorithms enumerate all minimal dominating sets of G. Note that whether a given set is a minimal dominating set can easily be checked in polynomial time. Each of our branching algorithms naturally defines a search tree, whose root corresponds to the input instance $(G, \emptyset)$, and every generated set in $\mathcal{D}$ appears in a leaf. By determining an upper bound on the number of leaves of the search tree, we also obtain an upper bound on the number of minimal dominating sets in G.

For the analysis of the running time of our algorithms, if at each branching step we make t new subproblems, where the measure of the instance is decreased by $c_1, c_2, \ldots, c_t$ in each respective subproblem, we obtain in the standard way a recurrence $T(n) \leq T(n - c_1) + T(n - c_2) + \ldots + T(n - c_t)$ for the running time. Such a recurrence is said to have *branching vector* $(c_1, c_2, \ldots, c_t)$. Then the number of nodes in the search tree, and hence the running time of the algorithm, is upper bounded by $O^*(\alpha^n)$, where α is the unique positive real root of $x^n - x^{n-c_1} - \ldots - x^{n-c_t} = 0$ [9]. The number α is called the *branching number* of this branching vector. It is common to round α to the fourth digit after the decimal point. By rounding the last digit up, we can use O-notation instead of O^*-notation [9]. If different branching rules are applied and different branching vectors are involved at different steps of an algorithm, then the branching vector with the highest branching number gives the upper bound we use. We will not do the calculations of α explicitly, but just say, e.g., that branching vector $(2, 2)$ has branching number 1.4143.

3 Chordal Graphs

A *chord* of a cycle is an edge between two non-consecutive vertices of the cycle. A graph is *chordal* if every cycle of length at least 4 has a chord. A vertex v is called *simplicial* if $N(v)$ is a clique. Every chordal graph has a simplicial vertex [15]. A *perfect elimination ordering (peo)* of a chordal graph G is an ordering $\langle v_1, \ldots, v_n \rangle$ of its vertices such that for every $1 \leq i \leq n$, the neighbors of v_i that appear after v_i in the ordering form a clique, i.e., v_i is a simplicial vertex in the graph $G[\{v_i, v_{i+1}, \ldots, v_n\}]$. It is well known that a graph is chordal if and only if it has a peo [11], and that a peo of a chordal graph can be computed in linear time [32].

Observe that the graphs H_n and S_n, defined in Section 2, are chordal graphs, giving us examples of chordal graphs with $3^{n/3} \approx 1.4422^n$ minimal dominating sets. The next theorem provides an upper bound on the number of minimal dominating sets in a chordal graph.

Theorem 1. *Every chordal graph on n vertices has at most 1.6181^n minimal dominating sets, and these can be enumerated in time $O(1.6181^n)$.*

Proof. Let G be a chordal graph on n vertices. We describe an algorithm that enumerates all minimal dominating sets of G. Our algorithm begins by computing a peo $\sigma = \langle v_1, \ldots, v_n \rangle$ of G. At each of the subsequent steps of the algorithm, the subproblem at hand is described by (G', D), where G' is an induced subgraph of G and $D \subseteq V(G) \setminus V(G')$. As long as $V(G')$ is non-empty, the algorithm performs several reduction and branching rules that will be described below. The measure of an instance (G', D) is the number of vertices in G'. Throughout the algorithm, the following invariant will be true for every instance (G', D):

Invariant 1: Let x be the vertex of G' that appears first in σ, i.e., $x = v_i$, where i is the smallest index of a vertex of G' in σ.

- *If x is dominated, then all vertices of $V(G) \setminus V(G')$ are dominated.*
- *If x is undominated, then every undominated vertex of $V(G) \setminus V(G')$ is adjacent to every vertex of $N_{G'}[x]$.*

We will prove Invariant 1 by induction on $n - |V(G')|$, which is the number of vertices that have been deleted from G. Invariant 1 trivially holds for the original instance $(G, \emptyset)$, when $n - |V(G')| = n - |V(G)| = 0$. As will become clear from the description of the algorithm below, the algorithm deletes at least one vertex from G' at every step, thereby increasing $n - |V(G')|$ by at least 1. Whenever the algorithm deletes a vertex from G' that is not yet dominated by D, we will argue why Invariant 1 is still true after this vertex is deleted, assuming the invariant was true before the vertex was deleted. Note that the invariant trivially remains true in case a dominated vertex is deleted from G'.

Let (G', D) be an instance of a subproblem, and let x be the vertex of G' that appears first in the peo of G, i.e., $x = v_i$ and every vertex v_j with $j < i$ belongs to $V(G) \setminus V(G')$. Note that x is a simplicial vertex of G' due to the definition of a peo and the fact that G' is an induced subgraph of G.

First suppose x is isolated in G'. If x is undominated, then adding x to D is the only way to dominate x. Hence we add x to D and delete x from G'. If x is dominated, then we delete x from G' without adding x to D. In order to show that the latter rule is safe, it suffices to show that there is no minimal dominating set D' of G that contains $D \cup \{x\}$ as a subset. Suppose, for contradiction, that G does have such a minimal dominating set D'. Then x has a private neighbor x'. Since x is dominated by D, we cannot have $x = x'$. Moreover, since x has no neighbors in G', vertex x' must belong to $V(G) \setminus V(G')$. But then x' is dominated by D due to Invariant 1, contradicting the assumption that x' is a private neighbor of x. Note that the above two rules are reduction rules, and no branching is involved. Since in both cases x is dominated by the time it is deleted from G', Invariant 1 clearly remains true.

From now on, we assume that x has at least one neighbor in G'. We take action depending on whether or not x is dominated. Note that exactly one of the two cases below applies; the corresponding rule will be executed by the algorithm.

Case 1: x is dominated. We branch on the choice of either adding x to D or discarding x.

- *Add x to D.* Since x was already dominated before it was added to D, it needs a private neighbor. As a result of Invariant 1, such a private neighbor of x must belong to $N_{G'}(x)$. Because x is simplicial, $N_{G'}(x)$ is a clique and this means that no vertex of $N_{G'}(x)$ can appear in a minimal dominating set containing D as a subset; note that the set D considered here is the set D after x has been added to it, and therefore D contains x. Consequently, we can safely delete $N_{G'}[x]$, which results in the instance $(G' - N_{G'}[x], D \cup \{x\})$, and gives a decrease of at least 2 vertices.
- *Discard x.* Since x is dominated, it is safe to simply delete x from G'. This results in the instance $(G' - x, D)$, and gives a decrease of 1 vertex.

Note that when Case 1 applies, every vertex in $V(G) \setminus V(G')$ is dominated. Since the algorithm in this case only deletes vertices from G' once they are dominated, Invariant 1 still holds when the algorithm picks the next vertex x.

Case 2: x is undominated. Let y be any neighbor of x in G'. We branch on the choice of either adding y to D or discarding y.

- *Add y to D.* Clearly, the set D (which now contains y) dominates x. We claim that x will never be part of a minimal dominating set containing D. After all, x would need a private neighbor, which does not exist since $N_{G'}[x] \subseteq N_{G'}[y]$ and every vertex in $V(G) \setminus V(G')$ is dominated by D as a result of Invariant 1. We can therefore safely delete both x and y from G', which results in the instance $(G' - \{x, y\}, D \cup \{y\})$, and gives a decrease of 2.
- *Discard y.* In this case, we simply delete y from G', which is safe for the following reason. Vertex x is not deleted and still needs to be dominated.

This can only be done if a vertex of $N_{G'}[x] \setminus \{y\}$ is added to D at a later step. Since x is simplicial, y is adjacent to every vertex of $N_{G'}[x] \setminus \{y\}$. Hence, when x becomes dominated, then so will y. Moreover, the fact that y is adjacent to every vertex of $N_{G'}[x]$ implies that Invariant 1 remains true after y is deleted. We obtain a new instance $(G' - y, D)$, which gives a decrease of 1.

Note that when Case 2 applies, x is undominated, so all undominated vertices in $V(G) \setminus V(G')$ are adjacent to $N_{G'}[x]$ due to Invariant 1. This implies that as soon as any vertex of $N_{G'}[x]$ is added to D, all vertices in $V(G) \setminus V(G')$ will be dominated. From the description of Case 2 it is clear that at the time x is deleted from G', at least one vertex of $N_{G'}[x]$ has been added to D. This means that when x is deleted from G', all vertices of $V(G) \setminus V(G')$ are dominated, including any neighbor y of x that was undominated at the moment it was deleted from G' during an execution of the second branch of Case 2. Hence, when the algorithm proceeds to the next vertex x, Invariant 1 is still true.

The algorithm continues to recurse as long as $V(G')$ is non-empty. For each instance (G', D) where $V(G')$ is empty, i.e., at each leaf of the search tree, the algorithm checks if D is a minimal dominating set of G. If so, it outputs D; otherwise, D is discarded. This check can clearly be done in polynomial time.

In the description of each case, we argued why the corresponding branching or reduction rule is safe. Hence we are guaranteed that for every minimal dominating set D' of G, the set D' will be generated in one of the branches of our algorithm. In order to analyze the running time of our algorithm, recall that the worst-case branching vector in Case 1 is $(2, 1)$, and the branching vector obtained in Case 2 is $(2, 1)$ as well. Branching vector $(2, 1)$ has branching number 1.61804. The measure of the original instance is n, and the measure strictly decreases during the application of any reduction or branching rule. This implies that the total number of nodes in the search tree, and hence the overall running time of the algorithm, is $O^*(1.61804^n) = O(1.6181^n)$.

To complete the proof, let us analyze the upper bound for $\mu(G)$. Let $L(n)$ be the number of leaves in the search tree of the above algorithm on input G. As mentioned previously, $\mu(G) \leq L(n)$. The algorithm only branches in Cases 1 and 2, and two new subproblems are created each time. For each node in the search tree, the number of leaves in the subtree rooted at this node is the sum of the numbers of leaves in the subtrees rooted at its two children. From the description of Cases 1 and 2, and since $L(n)$ is non-decreasing, it is clear that $L(n) \leq L(n-2)+L(n-1)$. We prove by induction that $L(n) \leq 1.6181^n$, for $n \geq 1$. Observe that $L(0) = 0 < 1$, $L(1) = 1 < 1.6181$, and $L(2) \leq 2 < 1.6181^2$. For the inductive step, we assume that $L(n-1) \leq 1.6181^{n-1}$ and $L(n-2) \leq 1.6181^{n-2}$, and we need to verify that $1.6181^{n-2} + 1.6181^{n-1} \leq 1.6181^n$, for $n \geq 3$. This amounts to observing that $1 + 1.6181 \leq 1.6181^2 = 2.61824761$. Consequently, $\mu(G) \leq L(n) \leq 1.6181^n$. □

For split graphs, a well known subclass of chordal graphs, we are able to give a better upper bound in the next section. In Section 5, we prove that the same

upper bound also holds for proper interval graphs, another well known subclass of chordal graphs.

4 Split Graphs

A graph $G = (V, E)$ is a *split graph* if V can be partitioned into a clique C and an independent set I, where (C, I) is called a *split partition* of G. A split partition can be computed in linear time [15], and is not necessarily unique. In particular, if a vertex $c \in C$ has no neighbors in I, then $(C \setminus \{c\}, I \cup \{c\})$ is also a split partition of G. Note that a minimal dominating set of G cannot contain a vertex $y \in I$ together with a neighbor of y.

The graph S_n, defined in Section 2, is a split graph. Hence there are split graphs with $3^{n/3} \approx 1.4422^n$ minimal dominating sets. We now prove an upper bound on the number of minimal dominating sets in a split graph that is significantly lower than the upper bound for chordal graphs given by Theorem 1.

Theorem 2. *Every split graph on n vertices has at most 1.4656^n minimal dominating sets, and these can be enumerated in time $O(1.4656^n)$.*

Proof. Let G be a split graph on n vertices. We describe an algorithm that enumerates all minimal dominating sets of G. The algorithm starts by computing a split partition (C, I) of G such that I is maximal, which can be done in linear time [15]. It then performs several branching and reduction rules. Just like in the algorithm in the previous section, the measure of an instance (G', D) is the number of vertices in G'. For each instance (G', D), we will use (C', I') to denote the split partition of G' with $C' \subseteq C$ and $I' \subseteq I$, which is uniquely defined by the split partition (C, I) of G that was computed earlier. Throughout the algorithm, the following invariant will be maintained:

Invariant 2: Every vertex of $I \setminus I'$ is dominated and no vertex of I' is dominated.

As long as $V(G')$ is non-empty, at least one of the four cases below applies. The algorithm performs the rule corresponding to the first applicable case, i.e., if it executes the rule described under Case j, then Case i does not apply for any $i < j$.

Case 1: There is a vertex $y \in I'$ with no neighbors in C'. Due to Invariant 2, adding y to D is the only way to ensure that y is dominated. We therefore add y to D and delete y from G', yielding the instance $(G - y, D \cup \{y\})$. This reduction rule decreases the measure of the instance by 1. Invariant 2 is clearly maintained.

Case 2: There is a vertex $x \in C'$ with at least two neighbors in I'. Let $y_1, y_2 \in I'$ be two neighbors of x. The algorithm branches on the choice of either adding x to D or discarding x.

- *Add x to D.* If we add x to D, then no vertex $y_i \in N_{G'}(x) \cap I'$ can belong to a minimal dominating set of G that contains D as a subset, since $N_{G'}[y_i] \subseteq$

$N_{G'}[x]$ implies that y_i would not have a private neighbor. Hence we can safely delete both x and every vertex $N_{G'}(x) \cap I'$ from G', yielding the instance $(G' - (\{x\} \cup (N_{G'}(x) \cap I')), D \cup \{x\})$. Note that Invariant 2 is true for this instance. Since x has at least two neighbors in I', the measure decreases by at least 3.

– *Discard x.* We delete x from G', which is safe for the following reason. Vertex y_1 is not deleted from G', and Invariant 2 tells us that y_1 is not yet dominated by D. This means that a vertex of $N_{G'}[y_1]$ will be added to D at a later step in order to dominate y_1; such a vertex will also dominate x. We obtain the instance $(G-x, D)$ and a measure decrease of 1. As Invariant 2 only concerns vertices of I, the invariant trivially remains true.

Case 3: There is a vertex $x \in C'$ with no neighbors in I'. Let us first prove that x is dominated. Recall the split partition (C, I) of G that was computed by the algorithm at the beginning. Since the independent set I was chosen to be maximal, every vertex of C, and vertex x in particular, originally had at least one neighbor y in I. Since y has been deleted from G' but x itself has not been deleted, this means that a neighbor $x' \in C$ of y must have been placed in D during an execution of the rule of Case 2. Note that x' dominates x. This implies that any minimal dominating set of G that contains D cannot contain x, as x would not have a private neighbor. The algorithm therefore safely discards x and deletes x from G', yielding the instance $(G' - x, D)$. The measure decreases by 1, and Invariant 2 clearly remains true.

Case 4: Every vertex in C' has exactly one neighbor in I'. We arbitrarily choose a vertex $y \in I'$. By Invariant 2, y is not dominated by D. In order to dominate y, we need to add either y or one of its neighbors in C' to D. Moreover, since every vertex in C' has exactly one neighbor on I' and all vertices in $I \setminus I'$ are dominated due to Invariant 2, every minimal dominating set of G that contains D also contains exactly one vertex of $N_{G'}[y]$. We therefore branch into the following $|N_{G'}(y)|+1$ subproblems: for every neighbor x of y, add x to D and delete y and all its neighbors from G', and, in the last subproblem, add y to D and delete y and all its neighbors from G'. This yields $j+1$ instances whose measure is $j+1$ smaller than the measure of (G', D). Invariant 2 is clearly true for each of these instances.

The algorithm continues to recurse as long as $V(G')$ is non-empty. At each leaf of the search tree, i.e., for every instance (G', D) where $V(G')$ is empty, the algorithm checks if D is a minimal dominating set of G. It outputs D if it is a minimal dominating set of G, and discards D otherwise. From the description of the algorithm, it is clear that for every minimal dominating set D' of G, the set D' will appear at one of the leaves of our algorithm. This proves the correctness of our algorithm.

Let us analyze the running time of our algorithm. In Case 2, the measure decreases by at least 3 in the first subcase and by 1 in the second subcase. Hence, the worst-case branching vector for Case 2 is $(3, 1)$, and the corresponding

branching number is 1.46558. The only other case where the algorithm branches is Case 4. To determine the corresponding worst-case branching vector, suppose $y \in I'$ is the vertex that the algorithm chooses during an application of Case 4. Assuming that y has $j \geq 1$ neighbors, the algorithm creates $j+1$ subproblems as explained in the description of Case 4, and for each one the measure decreases by $j+1$. Hence, the corresponding branching vector is $(j+1, j+1, \ldots, j+1)$, where the component $j+1$ appears $j+1$ times, and $j \geq 1$. It is well-known that the highest corresponding branching number is 1.4423, which is obtained when $j = 2$ (see e.g. [9]). Using the same arguments as in the proof of Theorem 1, we conclude that the overall running time of our algorithm is $O^*(1.46558^n) = O(1.4656^n)$.

To give an upper bound on $\mu(G)$, we use arguments similar to those in the last paragraph of the proof of Theorem 1. In particular, we see from the analysis above that the number of leaves in the search tree satisfies $L(n) \leq L(n-3)+L(n-1)$ and $L(n) \leq c \cdot L(n-c)$, for $c \geq 2$. Furthermore, as before, we have $L(0) = 0$, $L(1) = 1$, $L(2) \leq 2$, and $L(3) \leq 3$. We will prove by induction that $L(n) \leq 1.4656^n$. Assume that this induction hypothesis is true for split graphs on at most $n-1$ vertices. Hence, we need to verify that $1.4656^{n-3} + 1.4656^{n-1} \leq 1.4656^n$, and $c \cdot 1.4656^{n-c} \leq 1.4656^n$. The first is verified by observing that $1 + 1.4656^2 = 3.14798336 < 1.4656^3 = 3.14808441$, and the second by observing that $c \leq 1.4656^c$, for all $c \geq 2$. Thus we conclude that $\mu(G) \leq 1.4656^n$. □

5 Proper Interval Graphs

A graph is a *proper* (or *unit*) *interval graph* if unit length intervals of the real line can be assigned to its vertices, such that two vertices are adjacent if and only if their corresponding intervals intersect [29]. Proper interval graphs, also called *indifference graphs*, are chordal. A *proper interval ordering* $\langle v_1, v_2, \ldots, v_n \rangle$ of a graph satisfies the following property: if $v_i v_j$ is an edge with $i < j$, then the vertices $v_i, v_{i+1}, \ldots, v_j$ form a clique. A graph is a proper interval graph if and only if it has a proper interval ordering, and such an ordering can be computed in linear time [25].

Observe that the graph H_n, defined in Section 2, is a proper interval graph, giving us examples of proper interval graphs with $3^{n/3} \approx 1.4422^n$ minimal dominating sets. We now show that the upper bound on the number of minimal dominating sets in a split graph also forms an upper bound on the number of minimal dominating sets in a proper interval graph. Even though the two bounds are the same, the proof of the next theorem requires different arguments than the proof of Theorem 2. In particular, we use vertex weights in order to analyze the running time of the enumeration algorithm described in the proof of Theorem 3 below.

Theorem 3. *Every proper interval graph on n vertices has at most 1.4656^n minimal dominating sets, and these can be enumerated in time $O(1.4656^n)$.*

Proof. Let G be a proper interval graph. We will describe a branching algorithm that enumerates all minimal dominating sets of G. Our algorithm first computes

a proper interval ordering $\sigma = \langle v_1, v_2, \ldots, v_n \rangle$ of the input graph G. Let (G', D) be an instance of a subproblem. Recall that a vertex of G' is *dominated* if it has a neighbor in D, and *discarded* if the algorithm has decided not to add the vertex to D. In the previous two algorithms, any vertex that was discarded at a certain step, was also deleted from G' in the same step. In this algorithm, a vertex that is discarded might or might not be deleted from G', depending on whether or not it is dominated by D; we will explain this in detail below. We call a vertex *forbidden* if it is discarded but not deleted from G'. All vertices of G' that are neither dominated nor forbidden are said to be *free*. In order to define the measure of an instance (G', D), we assign weights to the vertices of G' as follows: free and dominated vertices have weight 1, and forbidden vertices have weight 0. The measure of an instance (G', D) is the total weight of the vertices in G'. Initially, all vertices of the input graph G are free, and thus have weight 1. Hence the total weight of the input instance $(G, \emptyset)$ is n. Throughout the algorithm, the following invariant will be true for every instance (G', D):

Invariant 3: For any vertex v_j of G', the following two properties hold:

(i) If v_j is forbidden, then every vertex v_i of G' with $i < j$ is also forbidden.
(ii) If v_j is dominated, then every vertex v_i of G' with $i < j$ is also dominated.

At the start of the algorithm, all vertices are free and Invariant 3 is trivially true. When we make new subproblems, we will argue that the invariant is still true for each of the subproblems, assuming that is was true on the given instance, unless this is trivial. Our algorithm will only delete vertices from G' that are dominated. Consequently, the following additional invariant will be trivially true for every instance (G', D): all vertices in $V(G) \setminus V(G')$ are dominated by D. This means in particular that a vertex in G' is never needed to dominate a vertex outside of G', and hence no vertex of G' can have a private neighbor outside of G'. In other words, given an instance (G', D) and a vertex v of G', if there exists a minimal dominating set D' of G containing $D \cup \{v\}$ as a subset, then the private neighbor of v belongs to $V(G')$.

Given an instance (G', D) of a subproblem, the algorithm chooses a vertex x of G' with the smallest index in σ, i.e., $x = v_i$ and every vertex v_j with $j < i$ belongs to $V(G) \setminus V(G')$. Observe that the relative ordering of the vertices of G' according to σ is a proper interval ordering of G'. Hence, x is a simplicial vertex in G'.

We distinguish two main cases, depending on whether or not x is forbidden. Each case has subcases, depending on the number of neighbors of x in G'. Only one case applies, and the corresponding rule will be executed by the algorithm. Observe that any vertex which is dominated and forbidden can safely be deleted from the graph, since such a vertex cannot be added to D and is already dominated. Our algorithm will execute this reduction rule whenever possible. As a consequence, during the analysis below, we may assume that G' does not contain any vertices that are dominated and forbidden. In particular, we assume that any vertex of G' that is forbidden is undominated.

Case 1a: x is not forbidden and has at least two neighbors in G'. Notice that since x is not forbidden, by Invariant 3(i), none of its neighbors in G' is forbidden, since x is the vertex with the smallest σ index in G'. We branch on the possibilities of adding x to D or discarding x.

- *Add x to D.* After x is added to D, all vertices in $N_{G'}[x]$ are dominated. Since x needs a private neighbor and $N_{G'}(x)$ is a clique, no vertex of $N_{G'}(x)$ can appear in a minimal dominating set containing D as a subset. Consequently, we can safely delete $N_{G'}[x]$ from G', which results in the instance $(G' - N_{G'}[x], D \cup \{x\})$. Since $N_{G'}(x)$ contains at least two vertices, each of weight 1, and x itself has weight 1, the measure decreases by at least 3.
- *Discard x.* In this case, we simply forbid x. If x was already dominated, it can now be deleted. If not, the new instance is the same as before: (G', D). Since the weight of x has decreased from 1 to 0, the measure has decreased by 1. Since x had the lowest σ index of all vertices in G', Invariant 3(i) is clearly preserved.

Case 1b: x is free (not forbidden and undominated) and has exactly one neighbor in G'. Let y be the neighbor of x in G'. Vertex x is free, so y is also free as a result of Invariant 3. Since x is undominated and is only adjacent to y in G', either x or y needs to be added to D to ensure that x is dominated. We branch on these possibilities.

- *Add x to D.* In this case, y becomes dominated, and no minimal dominating set containing D as a subset can contain y, as x then would not have a private neighbor. Consequently, we can safely delete x and y from G' in this case. We get the instance $(G' - \{x, y\}, D \cup \{x\})$, and the measure decreases by 2.
- *Add y to D.* Now x becomes dominated, and it can never become a member of a minimal dominating set containing D, as it would then not have a private neighbor. We can safely delete x and y, and mark all the remaining neighbors of y as dominated. It follows from the definition of a proper interval ordering that Invariant 3(ii) is preserved. We get the instance $(G' - \{x, y\}, D \cup \{y\})$, with a measure decrease of 2.

Case 1c: x is not forbidden, is dominated, and has exactly one neighbor in G'. Let y be the neighbor of x in G'. Since x is not forbidden, Invariant 3 ensures that y is not forbidden either.

If y is dominated, then no minimal dominating set of G contains $D \cup \{x\}$ as a subset, as then x would not have a private neighbor. Hence, in this case we simply discard x and delete x from G', yielding the instance $(G - x, D)$ and a measure decrease of 1. This is a reduction rule, and no branching is involved. We now assume that y is undominated. We distinguish two subcases, depending on whether or not x is the only neighbor of y in G'.

First suppose that $|N_{G'}(y)| \geq 2$, i.e., x is not the only neighbor of y in G'. Then we branch as follows.

– *Add x to D.* In this case x needs a private neighbor and the only possible one is y. Since y is the private neighbor of x, no vertex of $N_{G'}[y] \setminus \{x\}$ can be added to D. Hence we mark all vertices in $N_{G'}[y] \setminus \{x\}$ as forbidden. Since y has become dominated by D, we can safely delete x and y from G', yielding the instance $(G' - \{x, y\}, D \cup \{x\})$. The measure decreases by at least 3, as the weight of at least one neighbor of y other than x decreased by 1 when it became forbidden.
– *Discard x.* In this case, x is already dominated and is now also forbidden. We therefore delete x, and the total weight decreases by 1.

Now suppose that $N_{G'}(y) = \{x\}$. We again branch on the possibilities of either adding x to D or discarding x, but we obtain different lower bounds on the decrease of the measure.

– *Add x to D.* Since y is the only possible private neighbor of x, we delete both x and y from G'. This results in the instance $(G' - \{x, y\}, D \cup \{x\})$ and a measure decrease of 2.
– *Discard x.* Since y is undominated and has no neighbors in G' apart from x, we must add y to D to ensure that y is dominated. Hence we add y to D, and delete x and y from G'. This yields the instance $(G' - \{x, y\}, D \cup \{y\})$ and decreases the measure by 2.

Case 1d: x is not forbidden and has no neighbors in G'. This case corresponds to a reduction rule and involves no branching. If x is dominated, then we cannot add x to D, as otherwise x would not have a private neighbor. Hence we discard x and delete it from G', yielding the instance $(G' - x, D)$. If x is not dominated, then we must add x to D to ensure it is dominated. We then delete x from G', yielding the instance $(G' - x, D \cup \{x\})$. The measure decreases by 1 in both cases.

Case 2a: x is forbidden and has at least two non-forbidden neighbors in G'. Since x is forbidden and undominated, one of the non-forbidden neighbors of x in G' must be added to D. Let $\langle y_1, y_2, \ldots, y_d \rangle$ be an ordering of the non-forbidden vertices in $N_{G'}(x)$ according to their ordering in σ, where $d \geq 2$. Observe that, since σ is a proper interval ordering, $N_{G'}[x] \subseteq N_{G'}[y_1] \subseteq N_{G'}[y_2] \subseteq \ldots \subseteq N_{G'}[y_d]$. This implies that no two vertices of $N_{G'}(x)$ can appear in a minimal dominating set of G together, since the one with the smallest neighborhood would not have a private neighbor. Hence, exactly one vertex of $N_{G'}(x)$ can and must be added to D.

We branch on the choice of the vertex y_i to be added to D. For each i between 1 and d, if y_i is added to D, then $N_{G'}[x]$ is dominated, and by the arguments above we can safely delete $N_{G'}[x]$ from G'. In addition, we mark the remaining vertices of $N_{G'}(y_i)$ as dominated. Invariant 3(ii) is preserved due to the fact that σ is a proper interval ordering. We obtain $d \geq 2$ new instances whose measure is d smaller than the measure of (G', D), since there are exactly d vertices of weight 1 in $N_{G'}[x]$; after all, each of the vertices $y_1, \ldots, y_d$ is free due to Invariant 3, and therefore has weight 1. This corresponds to the branching vector $(d, d, \ldots, d)$ of

length d. As we already saw at the end of the proof of Theorem 2, the highest branching number for branching vectors of this type is obtained when $d = 3$. Hence, we can take the branching vector to be $(3, 3, 3)$ for this case.

Case 2b: x is forbidden and has exactly one non-forbidden neighbor in G'. Let y be the only non-forbidden neighbor of x. Since x is undominated, y must be added to D. We can safely delete x and y from G', and mark all remaining neighbors of y in G' as dominated. This is a reduction rule, involving no branching. Note that when y is added to D, all forbidden neighbors of x –if any– become dominated; recall that our algorithm automatically deletes such vertices from G'.

Case 2c: x is forbidden and has no non-forbidden neighbors in G'. In this case, D cannot be extended to a dominating set for G. Hence we stop recursing and simply discard D.

When the recursion stops, the algorithm checks at each leaf of the search tree if D is a minimal dominating set of G. It outputs D if this is the case, and discards D otherwise. The correctness of the algorithm follows from its description. In order to analyze the running time, we observe that the branching vectors for Cases 1a, 1b, and 2a are $(3, 1)$, $(2, 2)$, and $(3, 3, 3)$, respectively. In Case 1c, the obtained branching vector is either $(3, 1)$ or $(2, 2)$. Out of all these branching vectors, $(3, 1)$ gives the largest branching number, namely 1.46558. As in the proof of Theorem 2, we conclude that the overall running time of the algorithm is $O(1.4656^n)$. For the number of leaves in the search tree, we get the same recurrences here as in the proof of Theorem 2, and hence 1.4656^n is an upper bound on the number of minimal dominating sets. □

6 Tight Combinatorial Bounds

In this section, we use combinatorial arguments to obtain upper bounds on the number of minimal dominating sets in cographs, trivially perfect graphs, threshold graphs, and chain graphs. These arguments can trivially be turned into algorithms for enumerating all minimal dominating sets, where the running time of each algorithm is within a polynomial factor of the proved upper bound for the corresponding graph class. For each of the classes studied in this section, we provide examples that contain the maximum possible number of minimal dominating sets, thereby showing that the proved upper bounds are tight.

6.1 Cographs

Cographs are of particular interest in the study of the maximum number of minimal dominating sets, as the only known examples of graphs with 1.5704^n minimal dominating sets are cographs. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The *disjoint union* of G_1 and G_2 is the graph $G_1 \uplus G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. The *join* of G_1 and G_2 is the graph $G_1 \bowtie G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{v_1 v_2 \mid v_1 \in$

$V_1, v_2 \in V_2\}$). A graph G is a *cograph* if it can be constructed from isolated vertices by disjoint union and join operations.

The graph G^*, depicted in Fig. 1, is a cograph. It has 6 vertices and 15 minimal dominating sets [7]. Any graph G_n^* on $n = 6k$ vertices consisting of k disjoint copies of G^* is a cograph containing $15^{n/6} \approx 1.5704^n$ minimal dominating sets. No example of a graph with 1.5705^n or more minimal dominating sets is known. We now prove that no *cograph* with 1.5705^n or more minimal dominating sets exists.

Theorem 4. *Every cograph on n vertices has at most $15^{n/6}$ minimal dominating sets, and these can be enumerated in time $O^*(15^{n/6})$. Furthermore, there are cographs with $15^{n/6}$ minimal dominating sets.*

Proof. The graph G_n^*, defined in the paragraph preceding Theorem 4, is a cograph with $15^{n/6}$ minimal dominating sets, so it remains to prove the upper bound. We do this by induction on the number of vertices.

Using a simple computer program that generates all cographs on at most 7 vertices and determines how many minimal dominating sets each of them contains, it can readily be verified that Theorem 4 holds for every $n \leq 7$: in particular, cographs on 1, 2, 3, 4, 5, 6, and 7 vertices have at most 1, 2, 3, 6, 9, 15, and 20 minimal dominating sets, respectively. Let G be a cograph on $n \geq 8$ vertices. By the definition of a cograph, there exist two subgraphs G_1 and G_2 of G such that $G = G_1 \uplus G_2$ or $G = G_1 \bowtie G_2$. Let $n_i = |V(G_i)|$ for $i = 1, 2$. Note that $n_1 + n_2 = n$ and $1 \leq n_i \leq n - 1$ for $i = 1, 2$.

If $G = G_1 \uplus G_2$, then by Observation 1, we have $\mu(G) = \mu(G_1) \cdot \mu(G_2) \leq 15^{n_1/6} \cdot 15^{n_2/6} = 15^{n/6}$. Suppose that $G = G_1 \bowtie G_2$. Then any minimal dominating set of G_1 dominates every vertex in G_2, and vice versa. This means that any minimal dominating set of G_1 is a minimal dominating set of G, and the same holds for any minimal dominating set of G_2. Since G is the complete join of G_1 and G_2, every minimal dominating set of G that contains more than one vertex from G_1 (respectively G_2) does not contain any vertex from G_2 (respectively G_1). This means that any minimal dominating set of G that is not a minimal dominating set of G_1 or G_2 is of the form $\{v_1, v_2\}$, where $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$. Hence $\mu(G) \leq \mu(G_1) + \mu(G_2) + n_1 n_2 \leq 15^{n_1/6} + 15^{n_2/6} + n_1 n_2 = 15^{n_1/6} + 15^{(n-n_1)/6} + n_1(n - n_1)$. We now show that for every fixed $n \geq 8$ and $1 \leq n_1 \leq n-1$, the function $15^{n_1/6} + 15^{(n-n_1)/6} + n_1(n-n_1)$ is maximum when $n_1 \in \{1, n-1\}$.

For ease of notation, let us define, for every fixed $n \geq 8$, the function $f_n(x) = c^x + c^{n-x} + x(n-x)$ on the domain $x \in [1, n-1]$, where $c = 15^{1/6}$. The second derivative of $f_n(x)$ with respect to x is the function $f_n''(x) = (\ln c)^2 \cdot (c^x + c^{n-x}) - 2$. Since the function $c^x + c^{n-x}$ is strictly convex on the interval $[1, n-1]$ and has a global minimum at $x = n/2$, we have $c^x + c^{n-x} \geq 2c^{n/2}$ for every $x \in [1, n-1]$. Hence $f_n''(x) = (\ln c)^2 \cdot (c^x + c^{n-x}) - 2 \geq 2(\ln c)^2 \cdot c^{n/2} - 2 > 0$, where the last inequality holds due to the assumption that $n \geq 8$. This implies that $f_n(x)$ is a convex function, attaining its maximum at the boundaries of its domain: $f_n(1) = f_n(n-1) = c + c^{n-1} + n - 1$.

Substituting $c = 15^{1/6}$ again, we get that $\mu(G) \leq 15^{1/6} + 15^{(n-1)/6} + n - 1$. Using induction on n, one can easily verify that $15^{1/6} + 15^{(n-1)/6} + n - 1 \leq 15^{n/6}$ for any $n \geq 8$, implying that $\mu(G) \leq 15^{n/6}$ for any $n \geq 8$. Since we already verified the validity of the inequality $\mu(G) \leq 15^{n/6}$ for all $n \leq 7$, we conclude that any cograph on n vertices has at most $15^{n/6}$ minimal dominating sets for any n. □

6.2 Trivially Perfect Graphs

Trivially perfect graphs form a subclass of cographs, and they have various characterizations [1, 15]. A graph G is *trivially perfect* if and only if each connected induced subgraph of G contains a universal vertex [33].

Theorem 5. *Every trivially perfect graph on n vertices has at most $3^{n/3}$ minimal dominating sets, and these can be enumerated in time $O^*(3^{n/3})$. Furthermore, there are trivially perfect graphs with $3^{n/3}$ minimal dominating sets.*

Proof. Observe that H_n, defined in Section 2, is a trivially perfect graph. Hence we have examples of trivially perfect graphs with exactly $3^{n/3}$ minimal dominating sets. It remains to show that $\mu(G) \leq 3^{n/3}$ for any trivially perfect graph on n vertices. We prove this upper bound by induction on n.

It can easily be verified that the upper bound holds for any trivially perfect graph on at most 3 vertices. Let G be a trivially perfect graph on $n \geq 4$ vertices. First suppose G is disconnected, and let $n_1, \ldots, n_t$ denote the number of vertices in the connected components of G. Note that $n_1 + \ldots + n_t = n$ and $n_i < n$ for $1 \leq i \leq t$. By our induction assumption and Observation 1, we have that $\mu(G) \leq 3^{n_1/3} \cdot \ldots \cdot 3^{n_t/3} = 3^{(n_1+\ldots+n_t)/3} = 3^{n/3}$. Now suppose G is connected. Then, by the aforementioned characterization of trivially perfect graphs [33], we know that G has a universal vertex u. The only minimal dominating set of G that contains u is $\{u\}$. Furthermore, the set of minimal dominating sets of G that do not contain u is exactly the set of minimal dominating sets of $G - u$, since u is dominated by every vertex of $G - u$. Recall that the class of trivially perfect graphs is closed under taking induced subgraphs. Hence, the graph $G - u$ is trivially perfect, and we can apply the induction hypothesis on $G - u$. Consequently, $\mu(G) = \mu(G - u) + 1 \leq 3^{(n-1)/3} + 1 \leq 3^{n/3}$, where the validity of the last inequality for all $n \geq 4$ can easily be verified using induction on n. This completes the proof of Theorem 5. □

6.3 Threshold Graphs

Threshold graphs form a subclass of trivially perfect graphs. Threshold graphs have several characterizations [1, 15, 26]. In particular, a graph G is a *threshold graph* if and only if it is a split graph and, for any split partition (C, I) of G, there is an ordering $\langle x_1, x_2, \ldots, x_k \rangle$ of the vertices of C such that $N[x_1] \subseteq N[x_2] \subseteq \ldots \subseteq N[x_k]$, and there is an ordering $\langle y_1, y_2, \ldots, y_\ell \rangle$ of the vertices of I such that $N(y_1) \supseteq N(y_2) \supseteq \ldots \supseteq N(y_\ell)$ [26]. Recall that $\omega(G)$ denotes the size of a maximum clique in a graph G.

Theorem 6. *Every threshold graph G on n vertices has exactly $\omega(G)$ minimal dominating sets, and these can be enumerated in $O(n^2)$ time.*

Proof. Let G be a threshold graph with split partition (C, I). We can assume C to be a clique of size $\omega(G)$, since otherwise I contains a vertex y that dominates C, in which case $(C \cup \{y\}, I \setminus \{y\})$ is a split partition of G where the clique $C \cup \{y\}$ has size $\omega(G)$. Suppose G has a minimal dominating set S that contains two vertices x_1 and x_2 of C. By definition, we either have $N[x_1] \subseteq N[x_2]$ or $N[x_2] \subseteq N[x_1]$; assume that $N[x_1] \subseteq N[x_2]$. Then $S \setminus \{x_1\}$ is a dominating set of G whose size is smaller than S, contradicting the minimality of S. This implies that any minimal dominating set of G contains at most one vertex of C. Let $C' \subseteq C$ be the set of vertices of C that have no neighbor in I. Since G is threshold and C is a maximum clique, C' is not empty. This means that every minimal dominating set of G contains at least, and therefore exactly, one vertex of C. For every vertex $x \in C$, the set $\{x\} \cup (I \setminus N[x])$ is the unique minimal dominating set of G that contains x. Hence G contains exactly $|C|$ minimal dominating sets. Using the inclusion relationship of the neighborhoods of the vertices in C, these minimal dominating sets can easily be enumerated in time linear in the sum of the cardinalities of the sets, i.e., $O(|C|\,|I|)$. □

6.4 Chain Graphs

In this section, we combine a study of structural properties of minimal dominating sets in chain graphs with combinatorial arguments to exactly determine the maximum number of minimal dominating sets in a chain graph on n vertices. A bipartite graph $G = (A, B, E)$ is a *chain graph* if there is an ordering $\sigma_A = \langle a_1, a_2, \ldots, a_k \rangle$ of the vertices of A such that $N(a_1) \subseteq N(a_2) \subseteq \cdots \subseteq N(a_k)$, as well as an ordering $\sigma_B = \langle b_1, b_2, \ldots, b_\ell \rangle$ of the vertices of B such that $N(b_1) \supseteq N(b_2) \supseteq \cdots \supseteq N(b_\ell)$ [34]; see Fig. 4 for an example of a chain graph. The orderings σ_A and σ_B together form a *chain ordering* of G. Note that if a chain graph G is disconnected, then at most one connected component of G contains edges. Chain graphs are closely related to threshold graphs, as a bipartite graph (A, B, E) is a chain graph if and only if turning either A or B into a clique results in a threshold graph [26].

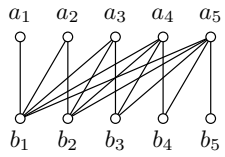

Fig. 4. The chain graph G_{10}, defined in the proof of Lemma 1. The orderings $\sigma_A = \langle a_1, a_2, \ldots, a_5 \rangle$ and $\sigma_B = \langle b_1, b_2, \ldots, b_5 \rangle$ together form a chain ordering of G_{10}.

It is an easy observation that in any (not necessarily chain) graph G, every maximal independent set of G is a minimal dominating set of G. More generally,

for any graph G, the minimal dominating sets of G that do not induce any edges are exactly the maximal independent sets of G. Hence, obtaining an upper bound on the number of maximal independent sets in a chain graph, which we do in the following lemma, will help us to obtain a tight upper bound on the number of minimal dominating sets in chain graphs (see Theorem 7 below).

Lemma 1. *Every chain graph on n vertices has at most $\lfloor n/2 \rfloor + 1$ maximal independent sets, and these can be enumerated in time $O(n^2)$. Furthermore, there are chain graphs that have $\lfloor n/2 \rfloor + 1$ maximal independent sets.*

Proof. Let $G = (A, B, E)$ be a chain graph on n vertices and assume, without loss of generality, that $|A| \leq \lfloor n/2 \rfloor$. Since any isolated vertex must belong to every maximal independent set, we may assume that G is connected. Let $\sigma_A = \langle a_1, a_2, \ldots, a_k \rangle$ and $\sigma_B = \langle b_1, b_2, \ldots, b_\ell \rangle$ be a chain ordering of G. Observe that b_1 dominates A and a_k dominates B. Let $\nu_i(G)$ be the number of maximal independent sets in G containing a_i, but not containing any vertex a_j with $j > i$. Consider $\nu_i(G)$ for some $i \in \{1, \ldots, k\}$. If every maximal independent set in G contains a vertex a_j with $j > i$, then $\nu_i(G) = 0$. Suppose there exists a maximal independent set S containing a_i, but not containing any vertex a_j with $j > i$. Clearly, none of the neighbors of a_i can be in S. Since σ_A and σ_B form a chain ordering of G, $N(a_p) \subseteq N(a_i)$ for every $p < i$. This means in particular that there is no edge between any vertex in $\{a_1, \ldots, a_{i-1}\}$ and any vertex in $B \setminus N(a_i)$. Hence the set $\{a_1, \ldots, a_i\} \cup B \setminus N(a_i)$ is the unique maximal independent set in G containing a_i and not containing any a_j with $j > i$. As a result, $\nu_i(G) \leq 1$ for every $i \in \{1, \ldots, k\}$. Note that B forms the only maximal independent set in G containing none of the vertices of A, and that every other maximal independent set in G contains at least one vertex from A. Since $\nu_i(G) \leq 1$ for every $i \in \{1, \ldots, k\}$ and $k \leq \lfloor n/2 \rfloor$ by assumption, we conclude that G has at most $\lfloor n/2 \rfloor + 1$ maximal independent sets. Clearly these sets can be listed in the described way in time that is linear in the sum of the cardinalities of the sets, i.e., $O(n^2)$.

For every even $n \geq 2$, let G_n be the chain graph obtained from two independent sets $A = \{a_1, \ldots, a_{n/2}\}$ and $B = \{b_1, \ldots, b_{n/2}\}$ by making a_i adjacent to every vertex in $\{b_1, \ldots, b_i\}$, for $i = 1, \ldots, n/2$; the graph G_{10} is depicted in Fig. 4. For every even $n \geq 2$, the graph G_n contains exactly $n/2 + 1 = \lfloor n/2 \rfloor + 1$ maximal independent sets, namely the set B, as well as $n/2$ sets of the form $\{a_1, \ldots, a_i\} \cup \{b_{i+1}, \ldots, b_{n/2}\}$, for $i \in \{1, \ldots, n/2\}$. Similarly, for every odd $n \geq 1$, a chain graph G_n on n vertices having exactly $\lfloor n/2 \rfloor + 1$ maximal independent sets can be obtained from the graph G_{n+1} defined above by deleting the vertex a_1. □

We now prove that no minimal dominating set of a chain graph G can induce more than one edge in G.

Lemma 2. *For every minimal dominating set S of a chain graph G, the graph $G[S]$ contains at most one edge.*

Proof. Let S be a minimal dominating set of a chain graph $G = (A, B, E)$. We first show that every connected component of $G[S]$ contains at most two vertices. Suppose, for contradiction, that $G[S]$ contains a connected component on more than two vertices. Since G is bipartite, this means that $G[S]$ contains an induced path on three vertices. Without loss of generality, let $a', a'' \in A$ and $b \in B$ be three vertices such that $\{a', b, a''\}$ induces a path on three vertices in $G[S]$. Note that b dominates both a' and a''. Hence, in order for a' and a'' to have private neighbors, there must exist vertices b' and b'' such that a' is the only vertex of S dominating b', and a'' is the only vertex of S dominating b''. This implies that there cannot be a chain ordering involving a and a', contradicting the assumption that G is a chain graph.

Let $\sigma_A = \langle a_1, \ldots, a_k \rangle$ and $\sigma_B = \langle b_1, \ldots, b_\ell \rangle$ form a chain ordering of G. Suppose $G[S]$ contains at least one edge, and let $a_i b_j$ be an edge of $G[S]$. We already showed that $G[S]$ does not contain an induced path on three vertices, so none of the vertices of $N(a_i) \setminus \{b_j\}$ and $N(b_j) \setminus \{a_i\}$ is in S. This means that if $G[S]$ contains an edge other than $a_i b_j$, then this edge is of the form $a_p b_q$ with $p < i$ and $q > j$, where $a_p \notin N(b_j)$ and $b_q \notin N(a_i)$. But then $N(a_p) \not\subseteq N(a_i)$, contradicting the assumption that σ_A is an ordering of the vertices of A such that $N(a_1) \subseteq \cdots \subseteq N(a_q) \subseteq \cdots \subseteq N(a_i) \subseteq \cdots \subseteq N(a_k)$. We conclude that every connected component of $G[S]$, apart from the component containing a_i and b_j, contains exactly one vertex. □

The following lemma is an easy consequence of Lemma 2.

Lemma 3. *Let ab be an edge of a chain graph $G = (A, B, E)$ with $a \in A$ and $b \in B$. If a or b has degree 1, then there is no minimal dominating set in G containing both a and b. If both a and b have degree at least 2, then there is exactly one minimal dominating set in G containing both a and b.*

From Lemmas 1, 2, and 3 we can readily deduce that every chain graph has at most $\lfloor n/2 \rfloor + m + 1$ minimal dominating sets. This bound is tightened below.

Theorem 7. *Every chain graph on $n \geq 2$ vertices and m edges has at most $\lfloor n/2 \rfloor + m$ minimal dominating sets, and these can be enumerated in $O(nm)$ time. Furthermore, there are chain graphs with $\lfloor n/2 \rfloor + m$ minimal dominating sets.*

Proof. Let $G = (A, B, E)$ be a chain graph on n vertices, and let $\sigma_A = \langle a_1, \ldots, a_k \rangle$ and $\sigma_B = \langle b_1, \ldots, b_\ell \rangle$ form a chain ordering of G. Without loss of generality, assume that $|A| \leq \lfloor n/2 \rfloor$. Since any isolated vertex must belong to every minimal dominating set, we may assume that G is connected. It is easy to check that G contains at most $\lfloor n/2 \rfloor + m$ minimal dominating sets in case $|A| = 1$ or $|B| = 1$. We therefore assume below that both A and B contain at least two vertices.

First suppose that at least one edge of G has an endpoint of degree 1. Then G has at most $m - 1$ minimal dominating sets S such that $G[S]$ contains an edge as a result of Lemmas 2 and 3. Every other minimal dominating set in G must be a maximal independent set in G, and G has at most $\lfloor n/2 \rfloor + 1$ such sets by Lemma 1. Hence G has at most $\lfloor n/2 \rfloor + m$ minimal dominating sets in this case.

Now suppose that for every edge of G both endpoints have degree at least 2. In particular, b_ℓ has degree at least 2, which means that b_ℓ is adjacent to a_{k-1}, which in turn implies that a_{k-1} is adjacent to every vertex in B. Recall that a_k also dominates B. Let S be a maximal independent set in G containing a_{k-1}. Since a_{k-1} dominates B, S does not contain any vertex of B. Since S is a maximal independent set of G, we must have $S = A$. In particular, $a_k \in S$. Let $\nu_i(G)$ be the number of maximal independent sets in G containing a_i, but not containing any vertex a_j with $j > i$. Note that $\nu_{k-1}(G) = 0$. As shown in the proof of Lemma 1, $\nu_i(G) \leq 1$ for every $i \in \{1, \ldots, k\}$. The set B is the only maximal independent set in G containing no vertices of A. Every other maximal independent set contains at least one vertex of A. The assumption that $|A| \leq \lfloor n/2 \rfloor$, together with $\nu_{k-1}(G) = 0$ and $\nu_i(G) \leq 1$ for every $i \in \{1, \ldots, k\}$, implies that G has at most $\lfloor n/2 \rfloor$ maximal independent sets, each of which forms a minimal dominating set in G. Due to Lemmas 2 and 3, G has exactly m minimal dominating sets S for which $G[S]$ contains an edge. Hence G contains at most $\lfloor n/2 \rfloor + m$ minimal dominating sets in total. Using the chain ordering, these sets can be enumerated in time linear in the sum of their cardinalities, i.e., $O(nm)$.

To show that the obtained upper bound is tight, we slightly adjust the example used in the proof of Lemma 1. For every even $n \geq 4$, let F_n be the chain graph obtained from two independent sets $A = \{a_1, \ldots, a_{n/2}\}$ and $B = \{b_1, \ldots, b_{n/2}\}$ by making a_i adjacent to every vertex in $\{b_1, \ldots, b_{i+1}\}$ for $i = 1, \ldots, n/2 - 1$, and by making $a_{n/2}$ into a false twin of $a_{n/2-1}$ by making $a_{n/2}$ adjacent to every vertex in B. Let $n \geq 4$ be an even integer. Since F_n has no vertices of degree 1, Lemma 3 implies that F_n has $m = |E(F_n)|$ minimal dominating sets that are *not* independent sets in F_n. Moreover, F_n has exactly $n/2 = \lfloor n/2 \rfloor$ minimal dominating sets that are maximal independent sets in F_n, namely the set B, as well each set of the form $\{a_1, \ldots, a_i\} \cup \{b_{i+2}, \ldots, b_{n/2}\}$ for $i = 1, \ldots, n/2 - 1$. Hence, F_n contains exactly $\lfloor n/2 \rfloor + m$ minimal dominating sets. For every odd $n \geq 5$, we can obtain a chain graph F_n with n vertices and m edges having exactly $\lfloor n/2 \rfloor + m$ minimal dominating sets by taking the graph F_{n-1} defined above and adding a new vertex b that is made adjacent to $a_{n/2}$ only. We point out that for every odd $n \geq 5$, the graph F_n has $m - 1$ minimal dominating sets that are *not* maximal independent sets, and $\lfloor n/2 \rfloor + 1$ minimal dominating sets that are maximal independent sets. □

7 Conclusions

We presented algorithms for enumerating all minimal dominating sets in several classes of graphs, yielding new upper bounds on the maximum number of minimal dominating sets in graphs belonging to these classes. The obtained bounds are significantly lower than the best known upper bound for general graphs. In a preliminary version of this paper, we also studied the maximum number of minimal dominating sets in cobipartite graphs: using purely combinatorial arguments, we obtained a lower bound of 1.3195^n and an upper bound of 1.5875^n [3].

While the lower bound of 1.3195^n is still the best one known, Liedloff recently improved the upper bound to 1.46^n using a branching algorithm [24].

The maximum number of minimal dominating sets in a general graph on n vertices is still unknown. It is conjectured that $15^{n/6}$, the best known lower bound, is indeed the correct answer [7]. Our results show that a counterexample to this conjecture cannot belong to any of the graph classes studied in this paper, with the exception of chordal graphs. A lower bound on the maximum number of minimal dominating sets in a bipartite graph is $6^{n/4} \approx 1.5650^n$, which is the number of such sets in the disjoint union of $n/4$ cycles of length 4. Is there an upper bound for bipartite graphs which is better than 1.7159^n? We conjecture that the maximum number of minimal dominating sets in proper interval graphs and in split graphs is $3^{n/3}$. As a sidenote, it is worth noticing that for most of the classes studied in this paper, the lower bound examples have many connected components. It might be interesting to investigate whether the same lower bounds still apply when a connectivity constraint is added.

As mentioned in the introduction, Fomin et al. [7] used their $O(1.7159^n)$ time algorithm for enumerating all minimal dominating sets in a graph to obtain an $O(2.8718^n)$ time algorithm for determining the domatic number of a graph. Could our enumeration algorithms be used to establish fast exact exponential-time algorithms for solving problems like Domatic Number or Connected Dominating Set on chordal graphs, or at least on split graphs? We point out that both problems are NP-complete on split graphs, and thus also on chordal graphs [22, 20].

Although, as a by-product, the algorithms presented in this paper always find a *minimum* dominating set in the input graph, we would like to point out that faster algorithms for this purpose exist in the literature. The fastest known algorithm for finding a minimum dominating set in a general n-vertex graph runs in time $O(1.4689^n)$ [17], while such a set can be found in time $O(1.3687^n)$ or in time $O(1.2252^n)$ if the input graph is chordal or split, respectively [30, 31]. For all other graph classes mentioned in Table 1, a minimum dominating set can be found in linear time; for cographs, trivially perfect graphs, threshold graphs, and chain graphs, one can apply the linear-time algorithm for permutation graphs due to Rhee et al. [28], while the linear-time algorithm for circular-arc graphs due to Chang [2] can be used to find a minimum dominating set in a proper interval graph.

As mentioned in Section 1, a related question that has generated a substantial amount of research is whether all minimal dominating sets in a graph can be enumerated in *output-polynomial time*, that is, by an algorithm whose running time is polynomial in the output size, i.e., the number of minimal dominating sets of the input graph [14, 18, 19]. For interval graphs and split graphs, such algorithms were obtained by Kanté et al. [19]. Our algorithms, except for the ones for threshold graphs and chain graphs, are not expected to be output-polynomial. Whether an output-polynomial algorithm exists for chordal graphs is an interesting open question.

References

1. A. Brandstädt, V. B. Le, and J. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications (1999).
2. M. S. Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. SIAM J. Comput. 27(6): 1671–1694 (1998).
3. J.-F. Couturier, P. Heggernes, P. van 't Hof, and D. Kratsch. Minimal dominating sets in graph classes: combinatorial bounds and enumeration. Proceedings of SOFSEM 2012, LNCS 7147, pp. 202–213 (2012).
4. J.-F. Couturier, P. Heggernes, P. van 't Hof, and Y. Villanger. Maximum number of minimal feedback vertex sets in chordal graphs and cographs. Proceedings of COCOON 2012, LNCS 7434, pp. 133–144 (2012).
5. D. Eppstein. Small maximal independent sets and faster exact graph coloring. J. Graph Algor. Appl. 7(2): 131–140 (2003).
6. F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica 52(2): 293–307 (2008).
7. F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov. Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. ACM Trans. Algorithms 5(1): 9:1–9:17 (2008).
8. F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, and Y. Villanger. Enumerating minimal subset feedback vertex sets. Proceedings of WADS 2011, LNCS 6844, pp. 399–410 (2011).
9. F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Springer, Texts in Theoretical Computer Science (2010).
10. F. V. Fomin and Y. Villanger. Finding induced subgraphs via minimal triangulations. Proceedings of STACS 2010, pp. 383–394 (2010).
11. D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. Pacific J. Math. 15: 835–855 (1965).
12. S. Gaspers and M. Mnich. Feedback vertex sets in tournaments. J. Graph Theory 72(1): 72–89 (2013).
13. P. A. Golovach, P. Heggernes, D. Kratsch, and R. Saei. An exact algorithm for subset feedback vertex set on chordal graphs. Proceedings of IPEC 2012, LNCS 7535, pp. 85-96 (2012).
14. P. A. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger. Generating all minimal edge dominating sets with incremental-polynomial delay. arXiv:1208.5345 (2012).
15. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Disc. Math. 57, Elsevier (2004).
16. T. W. Haynes and S. T. Hedetniemi (eds). *Domination in graphs*. Marcel Dekker Inc., New York (1998).
17. Y. Iwata. A faster algorithm for dominating set analyzed by the potential method. Proceedings of IPEC 2011, LNCS 7112, pp. 41–54 (2012).
18. M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. Enumeration of minimal dominating sets and variants. Proc. FCT 2011, LNCS 6914, pp. 298–394 (2011).
19. M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On the neighbourhood Helly of some graph classes and applications to the enumeration of minimal dominating sets. Proc. ISAAC 2012, LNCS 7676, pp. 289–298 (2012).
20. H. Kaplan and R. Shamir. The domatic number problem on some perfect graph families. Inform. Proc. Lett. 49(1): 51–56 (1994).

21. M. Krzywkowski. Trees having many minimal dominating sets. Inform. Proc. Lett. 113: 276–279 (2013)
22. R. C. Laskar and J. Pfaff. Domination and irredundance in split graphs. Technical Report 430, Clemson University (1983).
23. E. L. Lawler. A note on the complexity of the chromatic number problem. Inform. Proc. Lett. 5(3): 66–67 (1976).
24. M. Liedloff. Private communication (2012).
25. P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. Computers & Mathematics with Applications 25, pp. 15–25, 1993.
26. N. Mahadev and U. Peled. *Threshold graphs and related topics*. Annals of Discrete Mathematics 56, North Holland (1995).
27. J. W. Moon and L. Moser. On cliques in graphs. Israel J. Math. 3: 23–28 (1965).
28. C. J. Rhee, Y. D. Liang, S. K. Dhall, and S. Lakshmivarahan. An $O(n + m)$-time algorithm for finding a minimum-weight dominating set in a permutation graph. SIAM J. Computing 25(2): 404–419 (1996).
29. F. S. Roberts. Indifference graphs. In *Proof techniques in graph theory*, pp. 139–146, Academic Press, New York (1969).
30. J. M. M. van Rooij. Exact exponential-time algorithms for domination problems in graphs. University of Utrecht, PhD thesis, 2011.
31. J. M. M. van Rooij, J. Nederlof, and T. C. van Dijk. Inclusion/exclusion meets measure and conquer. Algorithmica, to appear. doi:10.1007/s00453-013-9759-2
32. D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. SIAM J. Computing 5(2): 266–283 (1976).
33. E. S. Wolk. The comparability graph of a tree. Proc. Amer. Math. Soc. 13: 789–795 (1962).
34. M. Yannakakis. Node deletion problems on bipartite graphs. SIAM J. Comput. 10: 310–327 (1981).