Minimal dominating sets in interval graphs and trees[☆]

Petr A. Golovach[a], Pinar Heggernes[a], Mamadou Moustapha Kanté[b], Dieter Kratsch[c,*], Yngve Villanger[a]

[a] *Department of Informatics, University of Bergen, N-5020 Bergen, Norway*
[b] *Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, Aubiére, France*
[c] *Université de Lorraine, LITA, Metz, France*

Abstract

We show that interval graphs on n vertices have at most $3^{n/3} \approx 1.4422^n$ minimal dominating sets, and that these can be enumerated in time $O^*(3^{n/3})$. As there are examples of interval graphs that actually have $3^{n/3}$ minimal dominating sets, our bound is tight. We show that the same upper bound holds also for trees, i.e. trees on n vertices have at most $3^{n/3} \approx 1.4422^n$ minimal dominating sets. The previous best upper bound on the number of minimal dominating sets in trees was 1.4656^n, and there are trees that have 1.4167^n minimal dominating sets. Hence our result narrows this gap. On general graphs there is a larger gap, with 1.7159^n being the best known upper bound, whereas no graph with 1.5705^n or more minimal dominating sets is known.

Keywords: Graphs, minimal dominating sets, graph classes, enumeration

1. Introduction

Enumerating vertex subsets of a graph satisfying a given property and establishing lower and upper bounds for the maximum number of such vertex subsets in graphs are central tasks in graph algorithms and combinatorics. Branching algorithms are one of the major techniques to design exact exponential algorithms solving NP-hard optimization problems [12]. Such recursive branching algorithms are also a major tool in constructing (exact exponential) enumeration algorithms. Furthermore their running time analysis can be used to obtain

[☆]This work is supported by the Research Council of Norway, the French National Research Agency (ANR project GraphEn / ANR-15-CE40-0009), and the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959.

[*]Corresponding author. Université de Lorraine, LITA, Metz, France. Phone: +33 387315444

Email addresses: petr.golovach@uib.no (Petr A. Golovach), pinar.heggernes@uib.no (Pinar Heggernes), mamadou.kante@isima.fr (Mamadou Moustapha Kanté), dieter.kratsch@univ-lorraine.fr (Dieter Kratsch), yngve.villanger@gmail.com (Yngve Villanger)

upper bounds on the maximum number of enumerated objects in a graph. A classical example is the famous result by Moon and Moser [26], which states that the maximum number of maximal independent sets in a graph on n vertices is $3^{n/3}$. Its proof can be translated into a branching algorithm that enumerates all maximal independent sets of a graph in time $O^*(3^{n/3})$, where the O^*-notation suppresses polynomial factors. The result is tight, as disjoint unions of triangles have exactly $3^{n/3}$ maximal independent sets. Triangle-free graphs, on the other hand, have at most $2^{n/2}$ maximal independent sets [19] and these can be enumerated in time $O^*(2^{n/2})$ [2]. Furthermore this bound is tight, as every 1-regular graph has $2^{n/2}$ maximal independent sets.

Recently there has been extensive research in this direction, both on general graphs and on graph classes, dealing with enumeration algorithms and combinatorial lower and upper bounds of minimal feedback vertex sets, minimal subset feedback vertex sets, minimal separators, and potential maximal cliques [4, 9, 11, 13, 14, 15]. Although the abovementioned results on maximal independent sets are tight, in general tight bounds are rare and there is often a gap between the best known upper bound and the best known lower bound, i.e., the largest number achieved by a known example. This is in particular the case for minimal dominating sets. Fomin, Grandoni, Pyatkin and Stepanov [10] gave an algorithm with running time $O(1.7159^n)$ 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. However, it is not known whether n-vertex graphs with 1.5705^n or more minimal dominating sets exist [10]. This gap has been narrowed on some well-known graph classes, like chordal graphs [3], trees [25], and cobipartite graphs [5]. Tight bounds have been obtained e.g., on cographs [3] and split graphs [5].

In this paper we study enumeration algorithms and lower and upper bounds for the maximum number of minimal dominating sets in interval graphs and trees. More precisely, we show that every interval graph on n vertices has at most $3^{n/3}$ minimal dominating sets which can be enumerated in time $O^*(3^{n/3})$. The bound is tight as a disjoint union of triangles, which is an interval (even a proper interval) graph, has exactly $3^{n/3}$ minimal dominating sets. Prior to our research reported in this paper, the best known upper bound on the number of minimal dominating sets in interval graphs was 1.6181^n, i.e. the best known upper bound for chordal graphs [3]. Proper interval graphs, which form a subset of interval graphs, have been known to have at most 1.4656^n minimal dominating sets, and hence our result closes the gap on that graph class as well. In addition we improve the upper bound on the number of minimal dominating sets of trees. Krzywkowski [25] has proved an upper bound of 1.4656^n on trees, and he gave a lower bound example with 1.4167^n minimal dominating sets. We show that trees have at most $3^{n/3} \approx 1.4422$ minimal dominating sets which can be enumerated in time $O^*(3^{n/3})$. Our results, together with known results on related graph classes, are summarized in Table 1.

Our study on the number of minimal dominating sets in graphs is motivated from various holds. First of all, domination in graphs is a very well studied subject with many applications in various fields, as can be seen by the number

Graph class	Lower bound	Previous upper bound	This paper
general	$15^{n/6}$	1.7159^n [10]	
chordal	$3^{n/3}$	1.6181^n [3]	
split	$3^{n/3}$	$3^{n/3}$ [5]	
interval	$3^{n/3}$	1.6181^n [3]	$3^{n/3}$
proper interval	$3^{n/3}$	1.4656^n [3]	$3^{n/3}$
tree	1.4167^n	1.4656^n [25]	$3^{n/3}$
co-bipartite	1.3195^n	1.4511^n [5]	
cograph	$15^{n/6}$	$15^{n/6}$ [3]	

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

of papers and books published on the subject, see e.g., [18]. Furthermore, the gap between the best known upper and lower bounds on the number of minimal dominating sets in general graphs naturally triggers curiosity about closing the gap on graph classes. Last but not least, enumeration of minimal dominating sets is strongly related to enumeration of minimal transversals of a hypergraph. The latter is a very well studied problem within the field of output polynomial algorithms. In fact, it is a long standing open question of great importance whether there is an output polynomial algorithm to enumerate the minimal transversals of a hypergraph, see e.g., [7, 8]. The existence of output polynomial algorithms to enumerate the minimal dominating sets has been studied on graph classes, and several works resolve it positively on some of the graph classes mentioned above [6, 7, 16, 20, 21, 22, 23].

2. Preliminaries

Graphs. 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, with $n = |V|$ 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 denoted by $N_G(v)$, and we let $N_G[v] = N_G(v) \cup \{v\}$. For a set $S \subseteq V$, we define analogously $N_G[S] = \bigcup_{v \in S} N_G[v]$ and $N_G(S) = N_G[S] \setminus S$. 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 $G[S]$. We use $G - v$ to denote the graph $G[V \setminus \{v\}]$, and $G - S$ to denote the graph $G[V \setminus S]$. A set $S \subseteq V$ is an *independent set* if $uv \notin E$ for every pair of vertices $u, v \in S$, and S is a *clique* if $uv \in E$ for every pair of vertices $u, v \in S$. A clique (independent set) is *maximal* if no proper superset of it is a clique (independent set).

Domination. A vertex set $D \subseteq V$ is a *dominating set* of G if $N[D] = V$. Every vertex v of a dominating set *dominates* the vertices in $N[v]$. A dominating set D is a *minimal dominating set* if no proper subset of D is a dominating set. If D is

a minimal dominating set then for every vertex $v \in D$, there is a vertex $x \in N[v]$ which is dominated only by v. We call such a vertex x a *private neighbor* of v, since x is not adjacent to any vertex in $D \setminus \{v\}$. To avoid confusion we may also call x a private neighbor of v *with respect to* D. Note that a vertex in D might be its own private neighbor. A vertex set $S \subseteq V$ is an *irredundant set* of G if every vertex of S has a private neighbor with respect to S. Observe that every subset of a minimal dominating set is irredundant. We denote the number of minimal dominating sets in a graph G by $\mu(G)$, and $\mu(n)$ denotes $\max\{\mu(G) : |V(G)| = n\}$. Since every minimal dominating set of G is the union of a minimal dominating set of each connected component of G, it is easy to see that $\mu(G) = \prod_{i=1}^{t} \mu(G_i)$, where G_1, G_2, ... , G_t are the connected components of G.

Finally let us mention that interval graphs and trees will be defined in the respective sections. All of the graph classes mentioned in the introduction can be recognized in linear time, and they are closed under taking induced subgraphs [1, 17] with the exception of trees.

3. Branching

The basics of branching algorithms, their time analysis and use in enumeration will be summarized in this section. For more details and examples, we refer to the book on "Exact exponential algorithms" [12] (Chapters 1, 2, 6).

Branching algorithms. A branching algorithm consists of reduction rules which modify the current instance and branching rules which branch into at least two new instances that we call subproblems. Typically the rules are given in a preference order and when the branching algorithm is called rcursively on an instance then it applies the first rule which can be applied. An execution of a branching algorithm is often illustrated by a search tree which is a rooted tree constructed as follows. To each node of the tree we assign an instance such that the input (instance) is assigned to the root. Whenever a branching rule is applied to an instance, then each subproblem is assigned to a child of the node of the instance. The running time of an execution is bounded by a polynomial in n times the number of leaves (making some usually fulfilled assumptions).

Measure. To analyse the running time of a branching algorithm, one assigns to every instance a real (often an integer) which is called its measure. In graphs this is typically done by assigning to each vertex of the instance a weight and then by taking the sum over the weight of all vertices as the measure of the instance. In our paper we use two types of measures. The first one, often called simple, is used in Section 4 and assigns to each vertex weight 1. The second one is used in Section 5, it is binary and assigns either 0 or 1 as weight to each vertex. It is worth mentioning that different measures may produce different upper bounds on the worst case running time of the very same algorithm. Choosing a sophisticated measure is often a crucial part of analyzing a branching algorithm.

Time analysis. To analyze the running time of a branching algorithm with respect to some fixed measure, one proceeds as follows. Let $L(n)$ be the maximum

number of leaves in a search tree corresponding to an execution of a branching algorithm on an input graph with n vertices. For any branching rule of the algorithm we determine a lower bound on the decreases of the measure from any original instance to all its subproblems obtained by the branching rule. Suppose for all $i = 1, 2, \ldots, t$, the measure is always decreased by at least c_i when branching to the i-th subproblem. This leads to a recurrence relation $L(n) = L(n - c_1) + L(n - c_2) + \ldots + L(n - c_t)$, which describes the worst case for this branching rule. This linear recurrence is said to have branching vector $(c_1, c_2, \ldots, c_t)$. To solve such a linear recurrence there are some important facts and tools, see e.g. [12]. To sketch the cruial ones, for the linear recurrences obtained via branching algorithms, the largest basic solution is of the form α^n where α is the unique positive real root of the characteristic polynomial $x^n - x^{n-c_1} - \ldots - x^{n-c_t} = 0$. Such a root α is called the branching number of the branching vector $(c_1, c_2, \ldots, c_t)$. Note that α is the largest root (real or complex) of the characteristic polynomial. One concludes that $L(n) \leq \alpha^n p(n)$ for all $n \geq 1$, where p is a polynomial in n. Therefore $L(n) = O^*(\alpha^n)$, assuming that only the branching rule with branching vector $(c_1, c_2, \ldots, c_t)$ is applied in the execution. Now such an analysis and computation has to be done for every branching rule and every branching vector. Let $\alpha_{\max}$ be the maximum over all branching numbers obtained. Now we may conclude that $L(n) \leq (\alpha_{\max})^n q(n)$ for all $n \geq 1$, where q is a polynomial in n. Consequently the corresponding branching algorithm has running time of $O^*((\alpha_{\max})^n)$.

Enumeration algorithms. Exponential-time algorithms solving enumeration problems are often branching algorithms. Those algorithms output the objects to be enumerated, e.g. the minimal dominating sets of the input graph in Sections 4 and 5, in the leaves of the search tree only. In every leaf they output at most one object, and hence the number of enumerated objects is bounded by the number of leaves in any execution and consequently an n-vertex graph has at most $L(n)$ objects. By running a polynomial time verification (if there is one) one might make sure that incorrect objects will not be outputted. Let us mention that such a verification algorithm for minimal dominating sets exists and runs in linear time. On the other hand, multiple occurences of an object at various leaves are allowed, and also leaves that do not output an object are allowed.

Upper bounds. If such an enumeration algorithm has a running time of $O^*(\alpha^n)$ then clearly it cannot enumerate more than $O^*(\alpha^n)$ objects, and thus an n-vertex graph has at most $O^*(\alpha^n)$ of these objects. Hence, in a trivial manner, the running time of a branching algorithm provides an upper bound for the number of enumerated objects in an n-vertex graph. Such an $O^*(\alpha^n)$ upper bound can often be strengthened to an upper bound of α^n (by removing the polynomial factor), i.e. for all $n \geq 1$ the number of enumerated objects in an n-vertex graph is at most α^n. Such a proof is done by induction proving the corresponding conjecture for all branching vectors obtained in the analysis of the branching algorithm.

4. Interval Graphs

A graph $G = (V, E)$ is an *interval graph* if there is a collection $\mathcal{I}$ of intervals of the real line and a bijection between V and $\mathcal{I}$ such that to each vertex $v \in V$ corresponds an interval $I_v = [\ell(v), r(v)] \in \mathcal{I}$, and two vertices u and w are adjacent in G if and only if $I_u \cap I_w \neq \emptyset$. Such a collection $\mathcal{I}$ is called an *interval model* of G. It is well known that every interval graph has a normalized interval model $\mathcal{I}$ in which all endpoints of intervals are pairwise disjoint and have their coordinates in $\{1, 2, \ldots, 2|V|\}$. We will thus assume in this section that every mentioned interval graph is accompanied with an interval model satisfying these conditions. In an interval graph, whenever we mention a set of vertices indexed with consecutive integers, like $U = \{u_1, u_2, \ldots, u_k\}$, it should be understood that $\ell(u_i) < \ell(u_{i+1})$ for $1 \leq i \leq k-1$. If U is an irredundant set, then it is easy to verify that this implies $r(u_i) < r(u_{i+1})$ for $1 \leq i \leq k-1$ [23].

Interval graphs are a well-known subclass of *chordal graphs*, which are the graphs that do not contain simple (chordless) cycles of length 4 or more as induced subgraphs. We refer to [1, 17] for more information on structural properties of interval graphs.

Regarding the number of minimal dominating sets in interval graphs, the best known lower bound example is a disjoint union of triangles, which has $3^{n/3}$ minimal dominating sets. No work addressing the maximum number of minimal dominating sets in interval graphs has been done prior to our work, however the upper bound 1.6181^n for chordal graphs [3] clearly also applies to interval graphs.

Recently Kanté et al. obtained a linear delay algorithm to enumerate the minimal dominating sets of interval graphs [23]. For this, they give structural properties of the minimal dominating sets of interval graphs with respect to the corresponding interval models. We show here how to use their results to prove an upper bound of $3^{n/3}$ on the number of minimal dominating sets of interval graphs.

Theorem 1 ([23])**.** *Let G be an interval graph, and let $D = \{u_1, u_2, \ldots, u_k\}$ be a set of vertices in G. Then D is a minimal dominating set of G if and only if the following conditions hold:*

1. *$\ell(u_1) < r(v)$ for every vertex v of G,*
2. *$\ell(v) < r(u_k)$ for every vertex v of G,*
3. *for every $i \in [1, k-1]$: if $\ell(v) < r(u_i)$ for every vertex v of G, then $i = k$, otherwise $\ell(w) < r(u_{i+1})$, where w is the vertex with the smallest $\ell(w)$ such that $\ell(w) > r(u_i)$,*
4. *for every $i \in [1, k-1]$: $r(y) < \ell(u_{i+1}) < r(z)$, where y is the private neighbor of u_i with the smallest $r(y)$, and z is the vertex with the smallest $r(z)$ such that $\ell(z) > r(u_i)$.*

We say that an irredundant set $U = \{u_1, \ldots, u_i\}$ in an interval graph is *good* if every vertex w satisfying $r(w) < \ell(u_i)$ has a neighbor in U. The following is a consequence of Theorem 1.

Corollary 1. *Let $D = \{u_1, u_2, \ldots, u_k\}$ be a minimal dominating set of an interval graph. Then $\{u_1, \ldots, u_i\}$ is a good irredundant set for every $1 \leq i \leq k$.*

From Theorem 1, it is clear that if we are given a good irredundant set $\{u_1, \ldots, u_i\}$ for some $i < k$, and we want to determine u_{i+1} such that there is a minimal dominating set containing $\{u_1, \ldots, u_i, u_{i+1}\}$, then it is enough to search for u_{i+1} among vertices whose intervals have their left endpoints between $r(y)$ and $r(z)$, where y and z are vertices defined by u_i as stated in the fourth condition of the theorem. This will be the main idea behind our result. The crucial point for proving our upper bound is the following.

Lemma 1. *Let $U = \{u_1, \ldots, u_i\}$ be a good irredundant set of an interval graph, and let W be the set of vertices w satisfying $r(y) < \ell(w) < r(z)$, where y is the private neighbor of u_i with the smallest $r(y)$, and z is the vertex with the smallest $r(z)$ such that $\ell(z) > r(u_i)$. Then every good irredundant set that contains U, contains at most one vertex of W.*

Proof. Let $U' = U \cup \{w, w'\}$ with $w, w' \in W$ and $w \neq w'$. We will prove that U' is not irredundant. Let $u = u_i$ and $\ell(w) < \ell(w')$. Recall that if $r(w) > r(w')$ then U' is not irredundant, thus we can assume that $r(w) < r(w')$. Let us distinguish three cases.

Case 1: Vertices w and w' are both adjacent to u. In this case both w and w' are dominated, and since $r(w) < r(w')$, w has no private neighbor with respect to U', thus U' is not irredundant.

Case 2: Neither w nor w' are adjacent to u. In this case, observe that $\ell(u) < \ell(w) < \ell(w') < r(z) \leq r(w) < r(w')$ by the definition of z and the previous discussion. Consequently, w and w' are adjacent, and w has no private neighbor with respect to U'. Hence U' is not irredundant.

Case 3: The remaining case is that w is adjacent to u, while w' is not adjacent to u. We argue that every neighbor of w is adjacent to either u or w'. Since w is dominated by u, this implies that w has no private neighbor, and thus U' is not irredundant. Assume for a contradiction that w has a neighbor z' that is not adjacent to u or w'. Then $r(u) < \ell(z') < r(z') < \ell(w')$. However, by the definition of z, this implies that $r(z) < \ell(w')$, which contradicts the definition of set W. □

We need a final lemma before we can give the main result of this section.

Lemma 2. *Let v be the vertex with the smallest $r(v)$ in an interval graph, and let X be the set of vertices x satisfying $\ell(x) < r(v)$. Then there is no good irredundant set that contains more than one vertex of X.*

Proof. Let $\{x, x'\} \subseteq U$, for $x, x' \in X$ and $x \neq x'$. We will show that U is not irredundant. Let $\ell(x) < \ell(x')$. As above, we can assume that $r(x) < r(x')$. Hence we have $\ell(x) < \ell(x') < r(v) \leq r(x) < r(x')$. Consequently, x and x' are adjacent, and they are both adjacent to v. (Note that x can be v.) Since by the definition of v, there is no vertex v' with $r(v') < \ell(x')$, every neighbor of x is adjacent to x'. Thus x has no private neighbor and hence U is not irredundant. □

Theorem 2. *There is a branching algorithm to enumerate all minimal dominating sets of an interval graph in time $O^*(3^{n/3})$.*

Proof. We describe a branching algorithm which enumerates the minimal dominating sets of a given interval graph $G = (V, E)$. The idea of the algorithm is to construct minimal dominating sets by passing the interval model 'from left to right", where the intervals are sorted in increasing order of their left endpoints. Hence in any subproblem we have a good irredundant set $\{u_1, u_2, \dots u_i\}$, $i \geq 0$, that we try to extend by adding a vertex, if it is not already a minimal dominating set.

An input to our branching algorithm, which we will also call a *subproblem*, is a pair of sets (U, Y), where $U \subseteq V$ is a good irredundant set, and $Y \subseteq V \setminus U$ is the set of vertices that are available for addition to U to obtain a minimal dominating set of G. The size of an instance, or subproblem, (U, Y) is $|Y|$. Initially the algorithm is called with input $(\emptyset, V)$, thus the size of the initial instance is n.

By Theorem 1 and Lemma 2, every minimal dominating set contains exactly one vertex from the set X described in Lemma 2. Note that set X is not empty, as it contains at least the vertex with the leftmost right endpoint. Hence, when called on input $(\emptyset, V)$, the algorithm branches into $|X|$ subproblems $(\{x\}, V \setminus X)$, one for every vertex $x \in X$. A set consisting of a single vertex is clearly a good irredundant set.

Assume now that the algorithm is called with input (U, Y), where $U = \{u_1, u_2, \dots, u_i\}$ is a good irredundant set. By Theorem 1 and Lemma 1, either U is a minimal dominating set, or every minimal dominating set that contains U must contain exactly one vertex of the set W that is described in Lemma 1. The algorithm first checks whether U is a dominating set. If so, the algorithm stops and outputs U; this corresponds to a leaf of the search tree. If U is not a dominating set, we branch into $|W|$ new subproblems $(U \cup \{w\}, Y \setminus W)$, one for every vertex w of W. If $U \cup \{w^*\}$ is not a good irredundant set for a $w^* \in W$, we can immediately discard the subproblem $(U \cup \{w^*\}, Y \setminus W)$, as it will never lead to a minimal dominating set of G.

In both of the cases above, we branch into a number t of subproblems for some $t \geq 1$, and the size of the instance is decreased by at least t in each subproblem. The corresponding branching vector $(t, t, \dots, t)$ containing t entries, is known to be maximum when $t = 3$. The branching number of the resulting branching vector $(3, 3, 3)$ is $3^{n/3}$. Due to Section 3, this immediately implies that the running time of the algorithm is $O^*(3^{n/3})$. □

Theorem 3. *The maximum number of minimal dominating sets in an interval graph on n vertices is $3^{n/3}$.*

Proof. Let $L(n)$ be the number of leaves in the search tree of the above algorithm on input G. As mentioned in Section 3, $\mu(G) \leq L(n)$. By the construction of the branching algorithm all branching vectors to consider are of the form $(t, t, \dots, t)$ containing $t \geq 2$ entries and one of them can always be applied (unless n is too small). 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 t children.

We claim that $\mu(G) \leq 3^{n/3}$ for any interval graph on n vertices. Let G be any interval graph on n vertices. Since $L(n)$ is non-decreasing it is clear that $L(n) \leq t \cdot L(n-t)$ for some $t \geq 2$. We prove by induction that $L(n) \leq 3^{n/3}$. Observe that $\mu(1) = L(1) < 3^{1/3} < 1.4423$. For the inductive step, we assume $L(n-t) \leq 3^{(n-t)/3}$ and we need to verify that $L(n) \leq 3^{n/3}$. Since $t \leq 3^{t/3}$ for all positive integers, we may deduce $L(n) \leq t \cdot L(n-t) \leq t \cdot 3^{(n-t)/3} \leq 3^n$ for all $t \geq 2$. Hence, whatever branching vector is applied in the algorithm to G, we have $\mu(G) \leq L(n) \leq 3^{n/3}$. This completes the proof.

Combined with the abovementioned lower bound of $3^{n/3}$ one obtains a tight bound of $3^{n/3}$ for the maximum number of minimal dominating sets in an interval graph. □

5. Trees

A graph is a *tree* if it is connected and acyclic. A graph is a *forest* if it is acyclic. Thus every induced subgraph of a tree is a forest. A vertex v of a tree T is a *leaf* if its degree is 1, and thus it has a unique neighbor. It is well-known that every tree on $n \geq 2$ vertices has at least two leaves. Recently Krzywkowski [25] obtained lower and upper bounds for the maximum number of minimal dominating sets in trees on n vertices. He provided lower bound examples for trees having 1.4167^n minimal dominating sets, and an upper bound of 1.4656^n. We improve the upper bound and narrow the gap between both bounds by showing that a tree on n vertices has at most $3^{n/3} \approx 1.4423^n$ minimal dominating sets.

Theorem 4. *There is a branching algorithm to enumerate all minimal dominating sets of a tree in time* $O^*(3^{n/3})$.

Proof. Like in the previous section, we prove this theorem by presenting a branching algorithm to enumerate all minimal dominating sets of a given tree $T = (V, E)$. An input to our algorithm, which we will again also call a *subproblem*, is a pair (T', D, F), where the forest T' is an induced subgraph of the input tree T, D is a subset of $V \setminus V(T')$ that dominates $V \setminus V(T')$, and F is a subset of $V(T')$. The algorithm will produce all minimal dominating sets of the original input tree T that are supersets of D but do not contain any vertex from F, by adding appropriate vertices of $V(T') \setminus F$ to D. Hence, initially the algorithm is called with input $(T, \emptyset, \emptyset)$.

At each step, some vertices of T' 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. Vertices that are added to D are immediately deleted from T'. Vertices that are discarded will be either deleted from T' or become *forbidden*, meaning that they will be added to F. We say that a vertex v of T' is *dominated* if D contains a vertex of $N_G[v]$, and *undominated* otherwise. Note that a dominated and forbidden vertex can safely be deleted from T' and F. It will be clear from the

description of the algorithm that no vertex of $V(T') \setminus F$ has a private neighbor in $V \setminus V(T')$. Hence, adding a vertex x of $V(T') \setminus F$ to D requires that x has a private neighbor in $V(T')$.

The size of an instance (T', D, F) is $|V(T') \setminus F|$, i.e., the number of vertices in T' that are not forbidden. Equivalently, we can say that forbidden vertices have weight 0, while all other vertices of T' have weight 1, and the size is the total weight of all vertices in T'. Initially none of the vertices are forbidden, and hence the size of the original input $(T, \emptyset, \emptyset)$ is n.

Now we describe the reduction and branching rules of the algorithm on input (T', D, F). The rules of the algorithm are given in the order of preference of their application. After each rule, we will argue for its safeness. Reduction Rules 0, 1, and 2 are trivially safe.

Reduction Rule 0: If T' contains a vertex v that is both dominated and forbidden, then delete v from T' and F, resulting in instance $(T' - v, D, F \setminus \{v\})$.

As a consequence of Reduction Rule 0, we can assume in the rest that no vertex of T' is both dominated and forbidden.

Reduction Rule 1: If T' contains an isolated vertex x which is not forbidden, then delete x from T'; if x was not dominated then add x to D, resulting in instance $(T' - x, D, F)$ or $(T' - x, D \cup \{x\}, F)$. If x is forbidden then the algorithm halts, i.e., no minimal dominating set containing D and not intersecting F is produced, since no such set exists as x cannot be dominated.

As a consequence of Reduction Rule 1, we can assume in the rest that T' has no isolated vertices. Our algorithm will then always branch on a vertex x which is a leaf. The unique neighbor of x is denoted by y. The neighbors of y different from x are denoted by $z_1, z_2 \ldots, z_t$, if any.

Reduction Rule 2: If T' has a forbidden leaf x whose unique neighbor y in T' is not forbidden, then add y to D, and delete x and y from T', resulting in instance $(T' - \{x, y\}, D \cup \{y\}, F \setminus \{x\})$. If both x and y are forbidden then the algorithm halts, i.e., no minimal dominating set containing D is produced, since no such set exists.

Thus from now on we may assume that T' has no forbidden leaves.

Reduction Rule 3: If T' has a dominated leaf x whose unique neighbor y is forbidden, and y has no other neighbor or all vertices in $N(y) \setminus \{x\} = \{z_1, z_2 \ldots, z_t\}$ are also forbidden, then add x to D, and delete x and y from T', resulting in instance $(T' - \{x, y\}, D \cup \{x\}, F \setminus \{y\})$.

To see that Reduction Rule 3 is safe, observe that this is the only way to ensure that y will become dominated under the given restrictions.

Reduction Rule 4: If T' has a dominated leaf x whose unique neighbor y in T' is also dominated, then delete x from T', resulting in instance $(T'-\{x\}, D, F)$.

To see that Reduction Rule 4 is safe, observe that if we added x to D then x would not have a private neighbor since y is already adjacent to some other vertex of D. The following rule is similar to Reduction Rules 0 and 1, and it is trivially safe.

Reduction Rule 5: If T' has an undominated leaf x whose unique neighbor y is forbidden, then add x to D and delete x and y from T', resulting in instance $(T' - \{x, y\}, D \cup \{x\}, F \setminus \{y\})$.

From now on we can assume that the unique neighbor of every undominated leaf is not forbidden.

Branching Rule 0: If x is an undominated leaf of T' with unique neighbor y and $N(y) \setminus \{x\} = \{z_1, z_2, \ldots, z_t\}$ possibly empty, then branch into two subproblems as follows:

(a) add x to D and delete x and y from T', resulting in instance $(T' - \{x, y\}, D \cup \{x\}, F)$;

(b) discard x: delete x and y from T' and add y to D, resulting in instance $(T' - \{x, y\}, D \cup \{y\}, F)$.

To see that Branching Rule 0 is safe, observe that either x or y must be added to D to ensure that x is dominated. However, they cannot both be added to D since then x would not have a private neighbor. Since neither x nor y is forbidden (by Reduction Rules 2 and 5), each branch gives a reduction of 2 in the instance size, and we get a branching vector $(2, 2)$ for Branching Rule 0.

Branching Rule 1: If x is a dominated leaf of T' whose unique neighbor y is not forbidden, and y has no neighbor other than x or all neighbors of y are forbidden then branch into two subproblems as follows:

(a) add x to D and delete x and y from T', resulting in instance $(T' - \{x, y\}, D \cup \{x\}, F)$;

(b) discard x: delete x and y from T' and add y to D, resulting in instance $(T' - \{x, y\}, D \cup \{y\}, F)$.

To see that Branching Rule 1 is safe, observe that y is not dominated by Reduction Rule 4. Thus y must be dominated, which can only be ensured by adding either x or y to D, since all other possible neighbors of y are forbidden. If x is added, y cannot be added since x would then not have a private neighbor. If y is added then x becomes forbidden and can be deleted since it is dominated. In this case y is either private for itself or it might have private

neighbors among its forbidden neighbors. Since neither x nor y is forbidden (by Reduction Rule 3), each branch gives a reduction of 2 in the instance size, and we get a branching vector $(2, 2)$ for Branching Rule 1.

Branching Rule 2: If x is a dominated leaf of T' whose unique neighbor y is not forbidden and $N(y) \setminus \{x\} = \{z_1, z_2, \ldots, z_t\}$, then let ℓ be the number of vertices in $\{z_1, z_2, \ldots, z_t\}$ that are not forbidden. By Branching Rule 1, we know that $t \geq \ell \geq 1$. Branch into subproblems as follows:

(a) add x to D, delete x and y from T', and make $z_1, z_2, \ldots, z_t$ forbidden, resulting in instance $(T' - \{x, y\}, D \cup \{x\}, F \cup \{z_1, \ldots, z_t\})$.

(b) discard x: if $\ell \geq 2$ then simply forbid x, resulting in instance $(T', D, F \cup \{x\})$, otherwise, if $\ell = 1$, then let z be the single vertex of $\{z_1, \ldots, z_t\}$ which is not forbidden, and branch further into the following subproblems:

  (b1) discard x and add y to D: delete $N[y]$ from T', resulting in instance $(T' - N[y], D \cup \{y\}, F \setminus N(y))$;

  (b2) discard x and discard y: add z to D, resulting in instance $(T - \{x, y, z\}, D \cup \{z\}, F)$;

To see that Branching Rule 2 is safe, observe first that y is undominated by our reduction rules. If we add x to D then y must become forbidden, since otherwise x would not have a private neighbor. For the same reason all other neighbors of y must also become forbidden, because if any of them entered D then y would cease to be a private neighbor of x. As y becomes forbidden and dominated it can be deleted. The other neighbors of y are simply added to F if they are not already there. This gives a reduction of $2 + \ell$ in the instance size. If $\ell \geq 2$, then the correctness of Branching Rule 2 follows, as we simply forbid x if we do not add it to D. This gives a branching vector $(4, 1)$. If $\ell = 1$, then discarding x means either discarding x and adding y, or discarding both x and y. In the first of these two last cases, when we add y to D, y needs a private neighbor, which cannot be x since x is already dominated, and hence z and y are the only possible private neighbors of y and thus z becomes forbidden together with x. Since after this, all neighbors of y are forbidden and dominated they can all be deleted, together with y. This gives a reduction of at least 3 in the instance size. If both x and y are discarded and thus forbidden, then z must be added to D, as this is the only way to dominate y. Consequently, all three vertices can be deleted, as x and y are both dominated and forbidden and z is in D. This again gives a reduction of 3 in the instance size. Together, the branches (a), (b1), and (b2) result in branching vector $(3, 3, 3)$ when $\ell = 1$.

Branching Rule 3: If x is a dominated leaf of T' whose unique neighbor y is forbidden with $N(y) \setminus \{x\} = \{z_1, z_2, \ldots, z_t\}$, then let ℓ be the number of vertices in $\{z_1, z_2, \ldots, z_t\}$ that are not forbidden. By Reduction Rule 3, we know that $t \geq \ell \geq 1$. Let $\ell = 1$ or $\ell \geq 3$. Branch into subproblems as follows:

(a) add x to D and forbid $z_1, z_2, \ldots, z_t$, resulting in instance $(T' - \{x, y\}, D \cup \{x\}, (F \setminus \{y\}) \cup \{z_1, \ldots, z_t\})$;

(b) discard x: If $\ell \geq 3$ then simply forbid x, resulting in instance $(T', D, F \cup \{x\})$. If $\ell = 1$ then let z be the single vertex in $\{z_1, z_2, \ldots, z_t\}$ which is not forbidden, and forbid x and add z to D, resulting in instance $(T' - \{y, z\}, D \cup \{z\}, (F \setminus \{y\}) \cup \{x\})$.

Let us argue that Branching Rule 3 is safe. If we add x to D then all other neighbors of y must be forbidden, since if y becomes dominated by one of them then x will not have a private neighbor. If we do not add x to D then we forbid it, in which case at least one other neighbor of y must be in D. If $\ell \geq 3$ we do not bother to identify such a neighbor of y, whereas when $\ell = 1$ there is only one forced choice. In every case, the vertices that are both dominated and forbidden are removed, and hence the correctness follows. If $\ell \geq 3$, we get branching vector $(4, 1)$ using branches (a) and (b). If $\ell = 1$ then we get branching vector $(2, 2)$, using branches (a) and (b).

Now while we can apply any of the Reduction Rules 0-5 or Branching Rules 0-3 we apply it. If none of them can be applied then each leaf x of T' is dominated, its unique neighbour y is forbidden and $N(y) \setminus \{x\}$ contains exactly two non forbidden vertices, and then we can apply Branching Rule 4 below.

Branching Rule 4: All leaves of T' are dominated and for each leaf x its unique neighbor y is forbidden and $N(y) \setminus \{x\}$ contains exactly two non forbidden vertices. Let x_0 be any leaf of T' and let y_0 be its unique neighbor. Now let x be a vertex of T' such that it has maximum distance to y_0 in T'. This implies that x is a leaf of T', is dominated and its unique neighbor y is forbidden. Moreover, $N(y) \setminus \{x\}$ contains precisely two non forbidden vertices, say z' and z''. Notice that at least one of $\{z', z''\}$, say z', is also a leaf of T'. Branch into subproblems as follows:

(a) add x to D and forbid z', z'', resulting in instance $(T' - \{x, y\}, D \cup \{x\}, (F \setminus \{y\}) \cup \{z', z''\})$;

(b) add z' to D and forbid x, z'', resulting in instance $(T' - \{z', y\}, D \cup \{z'\}, (F \setminus \{y\}) \cup \{x, z''\})$;

(c) add z'' to D and forbid x, z', resulting in instance $(T' - \{z'', y\}, D \cup \{z''\}, (F \setminus \{y\}) \cup \{x, z'\})$;

Let us argue that Branching Rule 4 is safe. Note that y is undominated by our reduction rules, and hence a neighbour of y must be added to D. Moreover, all the neighbors of y except x, z' and z'' are forbidden, and z' and x are already dominated. On the other hand, whenever two non forbidden neighbors of y are added to D then one of them is a dominated leaf and therefore would not have a private neighbor. Thus there are three possibilities to branch and in each one we add one of the three non forbidden neighbours of y to D, and forbid the

two others. In each case we remove the added vertex to D and remove also y that is dominated and forbidden, and since the vertices x, z' and z'' were not forbidden, we have a reduction of 3 in the instance size. Together, the branches (a), (b), and (c) result in branching vector $(3, 3, 3)$.

We have taken care of all the possibilities, and we can conclude that the algorithm is correct. For the upper bound on the running time and the maximum number of minimal dominating sets, observe first that every reduction rule reduces the size of an instance by at least 1. For the branching rules, let us study the branching vectors we have obtained and their branching numbers:
$(2, 2) \approx 1.4143$.
$(4, 1) \approx 1.3803$.
$(3, 3, 3) \approx 1.4423$.
The largest branching number is the one of $(3, 3, 3)$ which is exactly $3^{1/3}$ and thus the running time of the branching algorithm is $O^*(3^{n/3})$. □

The proof of the following theorem is similar to the one of Theorem 3. We also point that the upper bound holds for forests as well.

Theorem 5. *The maximum number of minimal dominating sets in a forest on n vertices is at most $3^{n/3}$.*

Proof. First, we claim that $\mu(T) \leq 3^{n/3}$ for any tree T on n vertices. Let $L(n)$ be the number of leaves in the search tree of the above algorithm on input T. By the construction of the branching algorithm the only branching vectors to consider are $(2, 2)$, $(3, 3, 3)$ and $(4, 1)$. Since $L(n)$ is non-decreasing it is clear that $L(n) \leq 2 \cdot L(n-2)$, $L(n) \leq 3 \cdot L(n-3)$ or $L(n) \leq L(n-4) + L(n-1)$ depending which rule and branching vector is applied to T (unless T is small). We prove by induction on n that $L(n) \leq 3^{n/3}$. Observe that $\mu(1) = L(1) < 3^{1/3} < 1.4423$. For the inductive step, we assume $L(n-t) \leq 3^{(n-t)/3}$ for all t. We need to consider our three branching vectors. For $(2, 2)$, we have $L(n) \leq 2 \cdot L(n-2) \leq 2 \cdot 3^{(n-2)/3} \leq 3^{n/3}$ since $2 < 3^{2/3}$. For $(3, 3, 3)$, we have $L(n) \leq 3 \cdot L(n-3) \leq 3 \cdot 3^{(n-3)/3} \leq 3^{n/3}$. Finally for $(4, 1)$, we have $L(n) \leq L(n-4) + L(n-1) \leq 3^{(n-4)/3} + 3^{(n-1/3} \leq 4 \cdot 3^{(n-4)/3}$ since $4 < 3^{4/3} \approx 4.3267$. This proves the claim for trees.

To see that the claim holds for forests, it is sufficient to recall that if T is a forest with the components $T_1, \ldots, T_k$, then $\mu(T) = \mu(T_1) \cdot \ldots \cdot \mu(T_k)$. □

6. Conclusions

We presented new upper bounds for the number of minimal dominating sets in interval graphs and trees. The new bound for interval graphs is tight. It also improves the upper bound 1.4656^n for proper interval graphs, given in [3], to the tight bound $3^{n/3}$. Very recently and independently of our work Couturier et al. also obtained the tight bound for interval graphs [5]. The new bound for trees narrows the gap between lower and upper bound, however we conjecture that improvements of the upper bound and lower bound are still possible. By

combining the results in this paper with the results in [3, 5] when looking at the classes of the original paper by Couturier et al. [3], by now tight bounds have been found for all of them, except for chordal graphs.

Unfortunately, no progress has been made on the general case. 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 [10]. The results presented in this paper and in various predecessors [3, 5, 25] show that a counterexample to this conjecture cannot belong to any of the studied graph classes, with the exception of chordal graphs.

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 actual number of the minimal dominating sets of the input graph [16, 20, 21]. The major goal of the construction and analysis of enumeration algorithms in this field of research are polynomial delay enumeration algorithms. Recently Kanté et al. [23] presented algorithms to enumerate all minimal dominating sets of interval and permutation graphs with linear delay, and gave in [24] an algorithm that enumerates minimal dominating sets in line graphs with polynomial delay. Since the algorithms in [23, 24] used deeply the structural properties of the studied graph classes and our tight upper bound for interval graphs were based on their work, can we use similarly their works on permutation and line graphs to narrow the gaps between lower and upper bounds in permutation and line graphs? It is worth emphasizing that structural properties of the studied graph classes were crucial for the significant progress made in the last years.

An important graph class that is not studied in the two communities is the class of bipartite graphs, which can shed light to the general case. Indeed, the known 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?

References

[1] A. Brandstädt, V. B. Le, and J. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications (1999).

[2] J. M. Byskov. Enumerating maximal independent sets with applications to graph colouring. Operation Research Letters 32: 547–556 (2004).

[3] J.-F. Couturier, P. Heggernes, P. van 't Hof, and D. Kratsch. Minimal dominating sets in graph classes: combinatorial bounds and enumeration. Theor. Comput. Sci. 487: 82–94 (2013).

[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. Proc. COCOON 2012, LNCS 7434, pp. 133–144 (2012).

[5] J.-F. Couturier, R. Letourneur, and M. Liedloff. On the number of minimal dominating sets on some graph classes. Theor. Comput. Sci. 562: 634–642 (2015).

[6] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. SIAM J. Computing, 24: 1278–1304 (1995).

[7] T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. SIAM J. Computing, 32:514–537 (2003).

[8] K. Elbassioni, K. Makino, and I. Rauf. Output-sensitive algorithms for enumerating minimal transversals for some geometric hypergraphs. Proc. ESA 2009, LNCS 5757, pp. 143–154 (2009).

[9] 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).

[10] 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).

[11] F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, and Y. Villanger. Enumerating minimal subset feedback vertex sets. Algorithmica 69(1): 216–231 (2014).

[12] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Springer, Texts in Theoretical Computer Science (2010).

[13] F. V. Fomin and Y. Villanger. Finding induced subgraphs via minimal triangulations. Proc. STACS 2010, pp. 383–394 (2010).

[14] S. Gaspers and M. Mnich. Feedback vertex sets in tournaments. J. Graph Theory 72(1): 72–89 (2013).

[15] P. A. Golovach, P. Heggernes, D. Kratsch, and R. Saei. An exact algorithm for subset feedback vertex set in chordal graphs. J. Discrete Algorithms 26: 7–15 (2014).

[16] P. A. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger. Generating all minimal edge dominating sets with incremental-polynomial delay. Proc. ICALP (1) 2013, LNCS 7965, pp. 485–496 (2013).

[17] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Disc. Math. 57, Elsevier (2004).

[18] T. W. Haynes and S. T. Hedetniemi (eds). *Domination in graphs*. Marcel Dekker Inc., New York (1998).

[19] M. Hujter and Z. Tuza. The number of maximal independent sets in triangle-free grahs. SIAM J. Disc. Math. 6: 284–288 (1993).

[20] 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).

[21] 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).

[22] M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On the enumeration of minimal dominating sets and related notions. SIAM J. Discrete Math. 28(4): 1916–1929 (2014).

[23] M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, and T. Uno. On the enumeration and counting of minimal dominating sets in interval and permutation graphs. Proc. ISAAC 2013, LNCS 8283, pp 339–349 (2013).

[24] M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, and T. Uno. Polynomial delay algorithm for listing minimal edge dominating sets in graphs. Proc. WADS 2015, LNCS 9214, pp 446–457 (2015).

[25] M. Krzywkowski. Trees having many minimal dominating sets. Inform. Proc. Lett. 113: 276–279 (2013)

[26] J. W. Moon and L. Moser. On cliques in graphs. Israel J. Math. 3: 23–28 (1965).