Exact (Exponential) Algorithms for the Dominating Set Problem

Fedor V. Fomin[1,⋆], Dieter Kratsch[2], and Gerhard J. Woeginger[3]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway
fomin@ii.uib.no
[2] LITA, Université de Metz, 57045 Metz Cedex 01, France
kratsch@sciences.univ-metz.fr
[3] Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
g.j.woeginger@tue.nl

Abstract. We design fast exact algorithms for the problem of computing a minimum dominating set in undirected graphs. Since this problem is NP-hard, it comes with no big surprise that all our time complexities are exponential in the number n of vertices. The contribution of this paper are 'nice' exponential time complexities that are bounded by functions of the form c^n with reasonably small constants $c < 2$: For arbitrary graphs we get a time complexity of 1.93782^n. And for the special cases of split graphs, bipartite graphs, and graphs of maximum degree three, we reach time complexities of 1.41422^n, 1.73206^n, and 1.51433^n, respectively.

1 Introduction

Nowadays, it is common believe that NP-hard problems can not be solved in polynomial time. For a number of NP-hard problems, we even have strong evidence that they cannot be solved in sub-exponential time. For these problems the only remaining hope is to design exact algorithms with good exponential running times. How good can these exponential running times be? Can we reach 2^{n^2} for instances of size n? Can we reach 10^n? Or even 2^n? Or can we reach c^n for some constant c that is very close to 1? The last years have seen an emerging interest in attacking these questions for concrete combinatorial problems: There is an $O^*(1.2108^n)$ time algorithm for independent set (Robson [13]); an $O^*(2.4150^n)$ time algorithm for graph coloring (Eppstein [4]); an $O^*(1.4802^n)$ time algorithm for 3-Satisfiability (Dantsin & al. [2]). We refer to the survey paper [14] by Woeginger for an up-to-date overview of this field. In this paper, we study the *dominating set* problem from this exact (exponential) algorithms point of view.

Basic Definitions. Let $G = (V, E)$ be an undirected, simple graph without loops. We denote by n the number of vertices of G. The open *neighborhood* of a vertex v is denoted by $N(v) = \{u \in V : \{u, v\} \in E\}$, and the closed

⋆ F. Fomin is supported by Norges forskningsråd project 160778/V30.

J. Hromkovič, M. Nagl, and B. Westfechtel (Eds.): WG 2004, LNCS 3353, pp. 245–256, 2004.

neighborhood of v is denoted by $N[v] = N(V) \cup \{v\}$. The degree of a vertex v is $|N(v)|$. For a vertex set $S \subseteq V$, we define $N[S] = \bigcup_{v \in S} N[v]$ and $N(S) = N[S] - S$. The subgraph of G induced by S is denoted by $G[S]$. We will write $G - S$ short for $G[V - S]$. A set $S \subseteq V$ of vertices is a *clique*, if any two of its elements are adjacent; S is *independent*, if no two of its elements are adjacent; S is a *vertex cover*, if $V - S$ is an independent set.

Throughout this paper we use the so-called *big-Oh-star* notation, a modification of the big-Oh notation that suppresses polynomially bounded terms: We will write $f = O^*(g)$ for two functions f and g, if $f(n) = O(g(n)\text{poly}(n))$ holds with some polynomial $\text{poly}(n)$. We say that a problem is solvable in *sub-exponential* time in n, if there is an effectively computable monotone increasing function $g(n)$ with $\lim_{n\to\infty} g(n) = \infty$ such that the problem is solvable in time $O(2^{n/g(n)})$.

The Dominating Set Problem. Let $G = (V, E)$ be a graph. A set $D \subseteq V$ with $N[D] = V$ is called a *dominating set* for G; in other words, every vertex in G must either be contained in D or adjacent to some vertex in D. A set $A \subseteq V$ *dominates* a set $B \subseteq V$ if $B \subseteq N[A]$. The *domination number* $\gamma(G)$ of a graph G is the cardinality of a smallest dominating set of G. The *dominating set problem* asks to determine $\gamma(G)$ and to find a dominating set of minimum cardinality. The dominating set problem is one of the fundamental and well-studied classical NP-hard graph problems (Garey & Johnson [6]). For a large and comprehensive survey on domination theory, we refer the reader to the books [8, 9] by Haynes, Hedetniemi & Slater. The dominating set problem is also one of the basic problems in parameterized complexity (Downey & Fellows [3]); it is contained in the parameterized complexity class W[2]. Further recent investigations of the dominating set problem can be found in Albers & al. [1] and in Fomin & Thilikos [5].

Results and Organization of This Paper. What are the best time complexities for dominating set in n-vertex graphs that we can possibly hope for? Well, of course there is the trivial $O^*(2^n)$ algorithm that simply searches through all the 2^n subsets of V. But can we hope for a sub-exponential time algorithm, maybe with a time complexity of $O^*(2^{\sqrt{n}})$? Section 2 provides the answer to this question: No, probably not, unless some very unexpected things happen in computational complexity theory ... Hence, we should only hope for time complexities of the form $O^*(c^n)$, with some small value $c < 2$. And indeed, Section 3 presents such an algorithm with a time complexity of $O^*(1.93782^n)$. This algorithm combines a recursive approach with a deep result from extremal graph theory. The deep result is due to Reed [12], and it provides an upper bound on the domination number of graphs of minimum degree three.

Furthermore, we study exact exponential algorithms for the dominating set problem on some special graph classes: In Section 4, we design an $O^*(1.41422^n)$ time algorithm for split graphs, and an $O^*(1.73206^n)$ time algorithm for bipartite graphs. In Section 5, we derive an $O^*(1.51433^n)$ time algorithm for graphs of maximum degree three. Note that for these three graph classes, the dominating set problem remains NP-hard (Garey & Johnson [6], Haynes, Hedetniemi & Slater [9]).

2 A Negative Observation

We will show that the existence of a sub-exponential time algorithm for the dominating set problem would be highly unlikely. Our (straightforward) argument exploits the structural similarities between the dominating set problem and the vertex cover problem: "Given a graph, find a vertex cover of minimum cardinality".

Proposition 1. *Let $G = (V, E)$ be a graph. Let G^+ be the graph that results from G by adding for every edge $e = \{u, v\} \in E$ a new vertex $x(e)$ together with the two new edges $\{x(e), u\}$ and $\{x(e), v\}$.*

Then the graph G has a vertex cover of size at most k, if and only if the graph G^+ has a dominating set of size at most k.

Proposition 2. *(Johnson & Szegedy [11])*
If the vertex cover problem on graphs of maximum degree three can be solved in sub-exponential time, then also the vertex cover problem on arbitrary graphs can be solved in sub-exponential time.

Proposition 3. *(Impagliazzo, Paturi & Zane [10])*
If the vertex cover problem (on arbitrary graphs) can be solved in sub-exponential time, then the complexity classes SNP and SUBEXP satisfy SNP $\subseteq$ SUBEXP (and this is considered a highly unlikely event in computational complexity theory).

Now suppose that the dominating set problem is solvable in sub-exponential time. Take an instance $G = (V, E)$ of the vertex cover problem with maximum degree at most three, and construct the corresponding graph G^+. Note that G^+ has at most $|V| + |E| \leq 5|V|/2$ vertices; hence, its size is linear in the size of G. Solve the dominating set problem for G^+ in sub-exponential time. Proposition 1 yields a sub-exponential time algorithm for vertex cover in graphs with maximum degree at most three. Propositions 2 and 3 yield that SNP $\subseteq$ SUBEXP.

3 An Exact Algorithm for Arbitrary Graphs

In this section we present the main result of our paper. It is the first exact algorithm for the dominating set problem breaking the natural $\Omega(2^n)$ barrier for the running time: We present an $O^*(1.93782^n)$ time algorithm to compute a minimum dominating set on any graph. Our algorithm heavily relies on the following result of Reed to restrict the search space.

Proposition 4. *(Reed [12])*
Every graph on n vertices with minimum degree at least three has a dominating set of size at most $3n/8$.

In fact, we will tackle the following generalization of the dominating set problem: An input for this generalization consists of a graph $G = (V, E)$ and a subset $X \subseteq V$. We say that a set $D \subseteq V$ dominates X, if $X \subseteq N[D]$. The goal

is to find a dominating set D for X of minimum cardinality. (Obviously, setting $X := V$ yields the classical dominating set problem). We will derive an exact $O^*(1.93782^n)$ time algorithm for this generalization.

The algorithm is based on the so-called pruning the search tree technique. The idea is to branch into subcases and to remove all vertices of degree one and two, until we terminate with a graph with all vertices of degree zero or at least three. Denote by V' the set of all vertices of degree at least three in this final graph. Let $t = |V'|$ and let $G' = G[V']$. Then Proposition 4 yields that there exists some vertex set in G' with at most $3t/8$ vertices that dominates all vertices of G'; consequently, there exists also a dominating set for $X' = X \cap V'$ of size at most $3t/8$ in G'. We simply test all possible subsets with up to $3t/8$ vertices to find a minimum dominating set D' for X' in G'. By using Stirling's approximation $x! \approx x^x e^{-x}\sqrt{2\pi x}$ for factorials, and by suppressing some polynomial factors, we see that the number of tested subsets is at most

$$\binom{t}{3t/8} = \frac{(t)!}{(3t/8)!\,(5t/8)!} = O^*(8^t \cdot 3^{-3t/8} \cdot 5^{-5t/8}) = O^*(1.93782^t),$$

where $8/(3^{3/8} \cdot 5^{5/8})$ is approximately 1.9378192. This test can be done in time $O^*(\sum_{i=1}^{3t/8} \binom{t}{i}) = O^*(1.93782^t)$. Finally, we add all degree zero vertices of X to the set D' to obtain a minimum dominating set of G.

Now let us discuss the branching into subcases. While there is a vertex of degree one or two, we pick such a vertex, say v, and we recurse distinguishing four cases depending on the degree of v and whether $v \in X$ or not.

Case A: The Vertex v Is of Degree One and $v \in V - X$. In this case there is no need to dominate the vertex v and there always exists a minimum dominating set for X that does not contain v. Then a minimum dominating set for $X - \{v\}$ in $G - \{v\}$ is also a minimum dominating set for X in G, and thus we recurse on $G - \{v\}$ and $X - \{v\}$.

Case B: The Vertex v Is of Degree One and $v \in X$. Let w be the unique neighbor of v. Then there always exists a minimum dominating set for X that contains w, but does not contain v. If D' is a minimum dominating set for $X - N[w]$ in $G - \{v, w\}$ then $D' \cup \{w\}$ is a minimum dominating set for X in G, and thus we recurse on $G - \{v, w\}$ and $X - N[w]$.

We need the following auxiliary result.

Lemma 1. *Let v be a vertex of degree 2 in G, and let u_1 and u_2 be its two neighbors. Then for any subset $X \subseteq V$ there is a minimum dominating set D for X such that one of the following holds.*

(i) $u_1 \in D$ *and* $v \notin D$*;*
(ii) $v \in D$ *and* $u_1, u_2 \notin D$*;*
(iii) $u_1 \notin D$ *and* $v \notin D$*.*

Proof. If there exists a minimum dominating set D for X that contains u_1 then there exists a minimum dominating set D' for X that contains u_1 but not v. In fact, if $v \in D$, then $D' = (D - \{v\}) \cup \{u_2\}$ is a dominating set for X and

$|D'| \leq |D|$. Similarly, if there exists a minimum dominating set for X that contains u_2 then there exists a minimum dominating set for X that contains u_2 but not v.

Thus we are left with five possibilities how v, u_1, u_2 might show up in a minimum dominating set D for X: (a) $u_1, u_2, v \notin D$; (b) $v \in D$ and $u_1, u_2 \notin D$; (c) $u_1 \in D$ and $v, u_2 \notin D$; (d) $u_2 \in D$ and $v, u_1 \notin D$; (e) $u_1, u_2 \in D$ and $v \notin D$. Now (i) is equivalent to (c) or (e), (ii) is equivalent to (b), and (iii) is equivalent to (a) or (d). This concludes the proof. □

Now consider a vertex v of degree two. Depending on whether $v \in X$ or not we branch in different ways. Additionally, the search is restricted to those minimum dominating sets D satisfying the conditions of Lemma 1.

Case C: The Vertex v of Degree 2 and $v \in V - X$. Let u_1 and u_2 be the two neighbors of v in G. By Lemma 1, we can branch into three subcases for a minimum dominating set D:

(C.1): $u_1 \in D$ and $v \notin D$. In this case if D' is a minimum dominating set for $X - N[u_1]$ in $G - \{u_1, v\}$ then $D' \cup \{u_1\}$ is a minimum dominating set for X in G, and thus we recurse on $G - \{u_1, v\}$ and $X - N[u_1]$.

(C.2): $v \in D$ and $u_1, u_2 \notin D$. In this case if D' is a minimum dominating set for $X - \{u_1, u_2\}$ in $G - \{u_1, v, u_2\}$ then $D' \cup \{v\}$ is a minimum dominating set for X in G, and thus we recurse on $G - \{u_1, v, u_2\}$ and $X - \{u_1, u_2\}$.

(C.3): $u_1 \notin D$ and $v \notin D$. In this case a minimum dominating set for X in $G - \{v\}$ is also a minimum dominating set for X in G, and thus we recurse on $G - \{v\}$ and X.

Case D: The Vertex v Is of Degree 2 and $v \in X$. Let u_1 and u_2 denote the two neighbors of v in G. Again according to Lemma 1, we branch into three subcases for a minimum dominating set D:

(D.1): $u_1 \in D$ and $v \notin D$. In this case if D' is a minimum dominating set for $X - N[u_1]$ in $G - \{u_1, v\}$ then $D' \cup \{u_1\}$ is a minimum dominating set for X in G. Thus we recurse on $G - \{u_1, v\}$ and $X - N[u_1]$.

(D.2): $v \in D$ and $u_1, u_2 \notin D$. In this case if D' is a minimum dominating set for $X - \{u_1, v, u_2\}$ in $G - \{u_1, v, u_2\}$ then $D' \cup \{v\}$ is a minimum dominating set for X in G. Thus we recurse on $G - \{u_1, v, u_2\}$ and $X - \{u_1, v, u_2\}$.

(D.3): $u_1 \notin D$ and $v \notin D$. Then $v \in X$ implies $u_2 \in D$. Now we use that if D' is a minimum dominating set for $X - N[u_2]$ in $G - \{v, u_2\}$ then $D' \cup \{u_2\}$ is a minimum dominating set for X in G. Thus we recurse on $G - \{v, u_2\}$ and $X - N[u_2]$.

To analyse the running time of our algorithm we denote by $T(n)$ the worst case number of recursive calls performed by the algorithm for a graph on n vertices. Each recursive call can easily be implemented in time polynomial in the size of the graph passed to the recursive call. In cases A and B we have $T(n) \leq T(n-1)$, in case C we have $T(n) \leq T(n-1) + T(n-2) + T(n-3)$ and in case D we have $T(n) \leq 2 \cdot T(n-2) + T(n-3)$. Standard calculations yield that the worst behavior of $T(n)$ is within a constant factor of α^n, where α is the largest root

of $\alpha^3 = \alpha^2 + \alpha + 1$, which is approximately 1.8393. Thus $T(n) = O^*(1.8393^n)$. Therefore, the most time consuming part of the algorithm is the procedure of checking all subsets of size at most $3t/8$ where $t \leq n$. As already discussed, this can be performed in $O^*(1.93782^n)$ steps by a brute force algorithm.

Summarizing, we have proved the following theorem.

Theorem 1. *A minimum dominating set of a graph on n vertices can be computed in time $O^*(1.93782^n)$ time. (The base of the exponential function in the running time is $8/(3^{3/8} \cdot 5^{5/8}) \approx 1.9378192$.)*

4 Split Graphs and Bipartite Graphs

In this section we present an exponential algorithm for the minimum set cover problem obtained by dynamic programming. This algorithm will then be used as a subroutine in exponential algorithms for the NP-hard minimum dominating set problems on split graphs and on bipartite graphs.

Let X be a ground set of cardinality m, and let $T = \{T_1, T_2, \ldots, T_k\}$ be a collection of subsets of X. We say that a subset $T' \subseteq T$ *covers* a subset $S \subseteq X$, if every element in S belongs to at least one member of T'. A minimum set cover of (X, T) is a subset T' of T that covers the whole set X. The minimum set cover problem asks to find a minimum set cover for given (X, T). Note that a minimum set cover of X can trivially be found in time $O^*(2^k)$ by checking all possible subsets of T.

Lemma 2. *There is an $O(mk\,2^m)$ time algorithm to compute a minimum set cover for an instance (X, T) with $|X| = m$ and $|T| = k$.*

Proof. Let (X, T) with $T = \{T_1, T_2, \ldots, T_k\}$ be an instance of the minimum set cover problem over a ground set X with $|X| = m$. We present an exponential algorithm solving the problem by dynamic programming.

For every nonempty subset $S \subseteq X$, and for every $j = 1, 2, \ldots, k$ we define $F[S; j]$ as the minimum cardinality of a subset of $\{T_1, \ldots, T_j\}$ that covers S. If $\{T_1, \ldots, T_j\}$ does not cover S then we set $F[S; j] := \infty$.

Now all values $F[S; j]$ can be computed as follows. In the first step, for every subset $S \subseteq X$, we set $F[S; 1] = 1$ if $S \subseteq T_1$, and $F[S; 1] = \infty$ otherwise. Then in step $j + 1$, $j = 1, 2, \ldots, k - 1$, $F[S; j + 1]$ is computed for all $S \subseteq X$ in $O(m)$ time as follows:

$$F[S; j + 1] = \min\{F[S; j],\ F[S - T_{j+1}; j] + 1\}.$$

This yields an algorithm to compute $F[S; j]$ for all $S \subseteq X$ and all $j = 1, 2, \ldots, k$ of overall running time $O(mk\,2^m)$. In the end, $F[X; k]$ is the cardinality of a minimum set cover for (X, T). □

Now we shall use Lemma 2 to establish an exact exponential algorithm to solve the NP-hard minimum dominating set problem for split graphs. Let us recall that a graph $G = (V, E)$ is a split graph if its vertex set can be partitioned into a clique C and an independent set I.

Theorem 2. *There is an* $O(n^2\,2^{n/2}) = O^*(1.41422^n)$ *time algorithm to compute a minimum dominating set for split graphs.*

Proof. If G is a complete graph or an empty graph, then the dominating set problem on G is trivial. If $G = (V, E)$ is not connected, then all of its components are isolated vertices except possibly one, say $G' = (V', E)$. If D' is a minimum dominating set of the connected split graph G' then $D' \cup (V - V')$ is a minimum dominating set of G.

Thus we may assume that the input graph $G = (V, E)$ is a connected split graph with a partition of its vertex set into a clique C and an independent set I where $|I| \geq 1$ and $|C| \geq 1$. Such a partition can be found in linear time (Golumbic [7]). A connected split graph has a minimum dominating set D such that $D \subseteq C$: consider a minimum dominating set D' of G with $|D' \cap I|$ as small as possible; then a vertex $x \in D' \cap I$ can be replaced by a neighbor $y \in C$. $N[x] \subseteq N[y]$ implies that $D'' := (D' - \{x\}) \cup \{y\}$ is a dominating set, and either $|D''| < |D'|$ (if $y \in D'$), or $|D''| = |D'|$ and $|D'' \cap I| < |D' \cap I|$–both contradicting the choice of D'.

Let $C = \{v_1, v_2, \ldots, v_k\}$. For every $j \in \{1, 2, \ldots, k\}$ we define $T_j = N(v_j) \cap I$. Clearly, $D \subseteq C$ is a dominating set in G if and only if $\{T_i \colon v_i \in D\}$ covers I. Hence the minimum dominating set problem for G can be reduced to the minimum set cover problem for (I, T) with $|I| = n - k$ and $|T| = k$. For $k \leq n/2$ this problem can be solved in time $O(n\,2^k) = O(n\,2^{n/2})$. For $k > n/2$, by Lemma 2, the problem can be solved in time $O((n - k)k\,2^{n-k}) = O(n^2\,2^{n/2})$.

Thus a minimum dominating set of G can be computed in time $O(n^2\,2^{n/2})$. □

A modification of the technique used to prove Theorem 2, can be used to obtain faster algorithms for graphs with large independent set.

Theorem 3. *There is an* $O(nz \cdot 3^{n-z})$ *time algorithm to compute a minimum dominating set for graphs with an independent set of size* z*. In particular, there is an* $O(n^2 \cdot 3^{n/2}) = O^*(1.73206^n)$ *time algorithm to compute a minimum dominating set for bipartite graphs.*

Proof. Let $G = (V, E)$ be a graph with an independent set of size z. Note that such an independent set can be identified in $O^*(1.2108^n)$ time by the algorithm of Robson [13].

Let $R = V - I$ denote the set of vertices outside the independent set. In an initial phase, we fix for every subset $X \subseteq R$ some corresponding vertex set $I_X \subseteq I$ via the following three steps.

1. Determine $Y = I - N[X]$.
2. Compute a vertex set $Z \subseteq N[X] \cap I$ of minimum cardinality subject to $R - N[X] - N[Y] \subseteq N(Z)$.
3. Set $I_X = Y \cup Z$.

First, we observe that $Y \subseteq I$ and $Z \subseteq I$ yield $I_X \subseteq I$. Secondly, $I \subseteq Y \cup N[X]$ implies that I is dominated by $X \cup I_X$, and $R - N[X] - N[Y] \subseteq N(Z)$ implies that

R is dominated by $X \cup I_X$. Consequently, the set $X \cup I_X$ forms a dominating set for the graph G. Thirdly, we claim that among all dominating sets D for G with $D \cap R = X$, the dominating set $X \cup I_X$ has the smallest possible cardinality: Indeed, $D \cap R = X$ means that the vertices in $Y = I - N[X]$ can only be dominated, if they are contained in D; hence $Y \subseteq D$. Furthermore, the vertices in $R - N[X] - N[Y]$ must all be dominated through some vertices in $N[X] \cap I$; in the second step, we determine the smallest possible subset $Z \subseteq N[X] \cap I$ with this property. Summarizing, for finding a minimum dominating set for G, it is sufficient to look through all the 2^{n-z} sets $X \cup I_X$.

What is the time complexity of this approach? The only (exponentially) expensive step for determining the sets I_X is the computation of the sets Z. And this expensive step boils down to solving a set covering problem that consists of a ground set $R - N[X] - N[Y]$ with at most $|R - X| \leq n - z - |X|$ elements, and that consists of a collection of $|N[X] \cap I| \leq z$ subsets. By Lemma 2, such a set covering problem can be solved in $O(nz \cdot 2^{n-z-|X|})$ time. The overall time for solving all set covering problems for all subsets $X \subseteq R$ is proportional to $\sum_{k=1}^{n-z} \binom{n-z}{k} nz \cdot 2^{n-z-k}$. This yields an overall time complexity of $O(nz \cdot 3^{n-z})$. □

Note that for graphs with an independent set of size $z \geq 0.39782 \cdot n$, the running time of the algorithm in Theorem 3 is better than the running time of the algorithm for general graphs from Section 3.

5 Graphs of Maximum Degree Three

Computer experiments suggest that exact exponential algorithms like the trivial $O^*(2^n)$ time algorithm, or like our $O^*(1.93782^n)$ algorithm from Section 3 have the slowest running times for fixed values of n, if the input graphs have large domination numbers. One possible explanation is that the algorithm has to spend a lot of time on checking that no vertex subset of size $\gamma(G) - 1$ is dominating (even in case a true minimum dominating set is detected at an early stage). Since graphs of maximum degree three have high domination numbers, the algorithms for general graphs do not behave well on these graphs.

In this section, we design a better exact algorithm for graphs of maximum degree three, by using the pruning a search tree technique and a structural property of minimum dominating sets in graphs of maximum degree three provided in the following lemma.

Lemma 3. *Let $G = (V, E)$ be a graph of maximum degree three. Then there is a minimum dominating set D of G with the following two properties:*

(i) every connected component of $G[D]$ is either an isolated vertex, or an isolated edge, and

(ii) if two vertices $x, y \in D$ form an isolated edge in $G[D]$, then x and y have degree three in G, and $N(x) \cap N(y) = \emptyset$.

Proof. Let D be a minimum dominating set of G with the maximum number of isolated vertices in $G[D]$. If $G[D]$ has a vertex x of degree three, then $D - \{x\}$

is a smaller dominating set of G, which is a contradiction. Thus the maximum degree of $G[D]$ is two.

Assume $G[D]$ has a vertex y of degree two. If the degree of y in G is two, then $D-\{y\}$ is a smaller dominating set of G, a contradiction. Otherwise let z be the unique neighbor of y in G that is not in D. If $z \in N[D-\{y\}]$ then $D-\{y\}$ is a smaller dominating set of G, another contradiction. Finally, if $z \notin N[D-\{y\}]$ then $D_1 := (D \cup \{z\}) - \{y\}$ is another minimum dominating set in G with a larger number of isolated vertices in $G[D_1]$ than in $G[D]$. This contradiction concludes the proof of property (i).

To prove property (ii), let us first show that any two adjacent vertices $x, y \in D$ have degree three in G. For the sake of contradiction, assume that y has degree less than three in G. Clearly y cannot have degree one, otherwise $D-\{y\}$ is a dominating set, a contradiction. Suppose y has degree two, and let $z \neq x$ be the second neighbor of y. If $z \in N[D-\{y\}]$ then $D-\{y\}$ is a dominating set of smaller size than D, a contradiction. If $z \notin N[D-\{y\}]$, then $D_2 := (D-\{y\}) \cup \{z\}$ is a minimum dominating set in G with a larger number of isolated vertices in $G[D_2]$ than in $G[D]$, another contradiction.

Finally, we prove that $N(x) \cap N(y) = \emptyset$ in G. For the sake of contradiction, assume that $N(x) \cap N(y) \neq \emptyset$. If $N[x] \subseteq N[y]$ then $D-\{x\}$ is a dominating set, and if $N[y] \subseteq N[x]$ then $D-\{y\}$ is a dominating set. In both cases this contradicts our choice of D. Hence $N(x) = \{y, w, u\}$ with $N(x) - N(y) = \{w\}$ and $N(x) \cap N(y) = \{u\}$. If $w \in N[D-\{x\}]$ then $D-\{x\}$ is a dominating set, another contradiction. If $w \notin N[D-\{x\}]$ then $D_3 := (D-\{x\}) \cup \{w\}$ is a minimum dominating set in G with a larger number of isolated vertices in $G[D_3]$ than in $G[D]$, the final contradiction. □

Now we construct a search tree algorithm using the restriction of the search space guaranteed by Lemma 3, i.e. for a graph $G = (V, E)$ of maximum degree three only vertex sets $D \subseteq V$ satisfying the properties of of Lemma 3 have to be inspected. W.l.o.g. we assume that the input graph is connected.

Theorem 4. *There is a $O^*(1.51433^n)$ time algorithm to compute a minimum dominating set on graphs of maximum degree three. (The base of the exponential function in the running time is the largest real root $\alpha \approx 1.51433$ of $\alpha^6 = \alpha^3 + 2\,\alpha^2 + 4$.)*

Proof. The algorithm is based on the pruning a search tree technique. The idea is to branch into subcases until we obtain a graph of maximum degree two, and for such a graph a minimum dominating set can be computed in linear time since each of its connected components is either an induced path P_k ($k \geq 1$) or an induced cycle C_k ($k \geq 3$). In this way we obtain all minimum dominating sets of G satisfying the properties of Lemma 3

More precisely, the input graph $G = (V, E)$ and $D = \emptyset$ correspond to the root of the search tree. To each node of the search tree corresponds an induced subgraph $G[V']$ of G and a partial dominating set $D \subseteq V - V'$ of G already chosen to be part of the dominating set obtained in any branching from this node. To each leaf of the tree corresponds a subgraph $G[V']$ of maximum degree two. For

each node of the search tree to which a subgraph $G[V']$ of maximum degree three corresponds the algorithm proceeds as follows: It chooses a neighbour (called x below) of a vertex of degree three such that x has smallest possible degree; then it inspects x and branches in various subcases. Suppose $(G[V'], D)$ corresponds to a node of the search tree and that $G[V']$ has maximum degree two. Then a linear time algorithm will be invoked to find a minimum dominating set D' of $G[V']$, and thus $D \cup D'$ is a dominating set of G. Finally the algorithm chooses a smallest set among all dominating sets of G obtained in this way and outputs it as a minimum dominating set of G.

To show that this algorithm has running time $O^*(1.51433^n)$ we have to study its branching into subcases. We denote by $T(n)$ the worst case number of recursive calls performed by the algorithm for a graph on n vertices.

The algorithm will pick a vertex x of degree three at most once, and this can only happen at the very beginning and only if all vertices of the input graph have degree three. Thus this branching is of no interest for the analysis of the overall running time of our algorithm.

We shall distinguish two cases: x has degree one or x has degree two. For each case the algorithm chooses one or two vertices to be added to the partial dominating set D and recurses on some smaller induced subgraphs. Based on Lemma 3 each connected component of $G[D]$ can be supposed to be a K_1 or a K_2. (Note that our analysis deals with the subgraph $G[V']$ that corresponds to the current node of the search tree.)

Case 1: x Is a Vertex of Degree One in $G[V']$. Let y be a degree three neighbour of x. Let z_1 and z_2 be the other neighbours of y. Clearly there is a minimum dominating set of $G[V']$ not containing x, and thus we may choose $x \notin D$ and $y \in D$. This leaves two possible subcases for the choice of the vertices to be added to D.

Subcase 1.A: $y \in D$ Isolated Vertex in $G[D]$. We add y to the dominating set D and recurse on $G - N[y]$. Since y has degree three the number of recursive calls on this subcase is $T(n-4)$.

Subcase 1.B: $y, z_i \in D$, $i \in \{1,2\}$, isolated edge in $G[D]$. Then we may obtain 2 subcases as follows: Add y, z_i, $i \in \{1,2\}$, to D and recurse on $G - (N[y] \cup N[z_i])$. By property (ii) of Lemma 3, this requires that z_i has degree three, hence we remove 6 vertices and the number of recursive calls on this subcase is at most $2\,T(n-6)$.

In total, in Case 1 we obtain the recurrence $T(n) \leq T(n-4) + 2 \cdot T(n-6)$. Standard calculations yield that the worst behavior of $T(n)$ is within a constant factor of α^n. This α is the largest real root of $\alpha^5 = \alpha + 2$, which is approximately 1.26717. Thus $T(n) = O^*(1.26717^n)$.

Case 2: x Is a Vertex of Degree Two in $G[V']$. Let y_1 and y_2 be the neighbours of x. W.l.o.g. let y_1 be a degree three vertex.

Case 2.1: y_1 and y_2 Are Adjacent in $G[V']$. Then there is a minimum dominating set of $G[V']$ not containing x, and thus either y_1 or y_2 must be added to D.

Case 2.1.1: y_2 Has Degree Two. Hence w.l.o.g. $y_1 \in D$ and $y_2 \notin D$. Thus either y_1 is an isolated vertex in $G[D]$, or $y, z \in D$ where z is the third neighbour of y_1. Thus we obtain the recurrence $T(n) \leq T(n-4) + \cdot T(n-6)$. Thus $T(n) = O^*(1.15097^n)$, where $\alpha \approx 1.15097$ is the largest real root of $\alpha^5 = \alpha + 2$.

Case 2.1.2: y_2 Has Degree Three. For $i = 1, 2$, let z_i be the third neighbour of y_i. Then either $y_1 \in D$ or $y_2 \in D$ is an isolated verted in $G[V']$, or $y_i, z_i \in D$ is an isolated edge in $G[V']$. Then we recurse on $G - N[y_i]$ and remove 4 vertices, or we recurse on $G - (N[y_i] \cup N[z_i])$ and remove 6 vertices. Consequently we obtain the recurrence $T(n) \leq 2 \cdot T(n-4) + 2 \cdot T(n-6)$. Thus $T(n) = O^*(1.33015^n)$, where $\alpha \approx 1.33015$ is the largest real root of $\alpha^6 = 2\,\alpha^2 + 2$.

Case 2.2: y_1 and y_2 Are Not Adjacent in $G[V']$. Since x has degree two either $x \in D$ is an isolated vertex in $G[D]$ or $x \notin D$.

Case 2.2.1: y_2 Has Degree Two. Let z_{11} and z_{12} the other neighbours of y_1, and let z_2 be the other neighbour of y_2.

Subcase 2.2.1.A: $x \in D$ Isolated Vertex in $G[D]$. We add x to the dominating set D and recurse on $G - N[x]$. Since x has degree two the number of recursive calls on this subcase is $T(n-3)$.

Subcase 2.2.1.B: $y_i \in D$ Isolated Vertex in $G[D]$. For $i = 1, 2$, we add y_i to the dominating set D and recurse on $G - N[y_i]$. Since y_1 has degree three and y_2 has degree two, the number of recursive calls on this subcase is at most $T(n-3) + T(n-4)$.

Subcase 2.2.1.C: $y_1, z_{1j} \in D$, $j \in \{1, 2\}$, Isolated Edge in $G[D]$. Then we may obtain 2 subcases as follows: Add y_1, z_{1j}, $j \in \{1, 2\}$, to D and recurse on $G - (N[y_1] \cup N[z_{1j}])$. By property (ii) of Lemma 3, this requires that z_{1j} has degree three, hence we remove 6 vertices and the number of recursive calls on this subcase is at most $2\,T(n-6)$.

In total, in Case 2.2.1 we obtain the recurrence $T(n) \leq 2 \cdot T(n-3) + T(n-4) + 2 \cdot T(n-6)$. As we have seen before, the worst behavior of $T(n)$ is within a constant factor of α^n. This α is the largest real root of $\alpha^6 = 2\,\alpha^3 + \alpha^2 + 2$, which is approximately 1.48613. Thus $T(n) = O^*(1.48613^n)$.

Case 2.2.2: y_2 Has Degree Three. Let z_{11} and z_{12} be the other neighbours of y_1, and let z_{21} and z_{22} be the other neighbours of y_2.

Subcase 2.2.2.A: $x \in D$ Isolated Vertex in $G[D]$. We add x to the dominating set D and recurse on $G - N[x]$. Since x has degree two the number of recursive calls on this subcase is $T(n-3)$.

Subcase 2.2.2.B: $y_i \in D$ Isolated Vertex in $G[D]$. For $i = 1, 2$, we add y_i to the dominating set D and recurse on $G - N[y_i]$. y_1 and y_2 have degree three, thus the number of recursive calls on this subcase is at most $2\,T(n-4)$.

Subcase 2.2.2.C: $y_i, z_{ij} \in D$, $i, j \in \{1, 2\}$, Isolated Edge in $G[D]$. Then we may obtain 4 subcases as follows: Add y_i, z_{ij}, $i, j \in \{1, 2\}$, to D and recurse on $G - (N[y_{ij}] \cup N[z_{ij}])$. This requires that z_{ij} has degree three, hence we remove 6 vertices and the number of recursive calls on this subcase is at most $4\,T(n-6)$.

In total, in Case 2.2.2 we obtain the recurrence $T(n) \leq T(n-3)+2\cdot T(n-4)+4\cdot T(n-6)$. The worst behavior of $T(n)$ is within a constant factor of α^n, where α is the largest real root of $\alpha^6 = \alpha^3+2\,\alpha^2+4$, which is approximately 1.5143218. Thus $T(n) = O^*(1.51433^n)$. □

References

1. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. *Fixed parameter algorithms for dominating set and related problems on planar graphs.* Algorithmica 33, 2002, pp. 461–493.
2. E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. *A deterministic* $(2-2/(k+1))^n$ *algorithm for k-SAT based on local search.* Theoretical Computer Science 289, 2002, pp. 69–83.
3. R. G. Downey and M. R. Fellows. *Parameterized complexity.* Monographs in Computer Science, Springer-Verlag, New York, 1999.
4. D. Eppstein. *Small maximal independent sets and faster exact graph coloring.* Proceedings of the 7th Workshop on Algorithms and Data Structures (WADS'2001), LNCS 2125, Springer, 2001, pp. 462–470.
5. F. V. Fomin and D. M. Thilikos. *Dominating sets in planar graphs: Branch-width and exponential speed-up.* Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA'2003), 2003, pp. 168–177.
6. M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W.H. Freeman and Co., San Francisco, 1979.
7. M. C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.
8. T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of domination in graphs.* Marcel Dekker Inc., New York, 1998.
9. T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Domination in graphs: Advanced Topics.* Marcel Dekker Inc., New York, 1998.
10. R. Impagliazzo, R. Paturi, and F. Zane. *Which problems have strongly exponential complexity?* Journal of Computer and System Sciences 63, 2001, pp. 512–530.
11. D. S. Johnson and M. Szegedy. *What are the least tractable instances of max independent set?* Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA'1999), 1999, pp. 927–928.
12. B. Reed. *Paths, stars and the number three.* Combinatorics, Probability and Computing 5, 1996, pp. 277–295.
13. J.M. Robson. *Algorithms for maximum independent sets.* Journal of Algorithms 7, 1986, pp. 425–440.
14. G. J. Woeginger. *Exact algorithms for NP-hard problems: A survey.* Combinatorial Optimization: "Eureka, you shrink", LNCS 2570, Springer, 2003, pp. 185–207.