Empirical Evaluation of Splitter-Game Based Algorithms for the Dominating Set Problem

Moritz Zielke
(359562)

Supervisors:
Roman Rabinovich & Sebastian Siebertz

Bachelor Thesis submitted

to

Prof. Dr. Stephan Kreutzer
Technische Universität Berlin
Institut für Softwaretechnik und theoretische Informatik
Lehrstuhl für Logik und Semantik

Prof. Dr. Uwe Nestmann
Technische Universität Berlin
Institut für Softwaretechnik und theoretische Informatik
Lehrstuhl für Modelle und Theorie verteillter Systeme

Berlin, May 30, 2016

Hiermit erkläre ich, dass ich die vorliege Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den

Zusammenfassung

Es wird ein Algorithmus vorgestellt, der das *dominating set* Problem (G, k) entscheidet und parametrisierbar ist für *nowhere dense* Klassen von Graphen. Der Algorithmus basiert auf dem Algorithmus von Grohe, Kreutzer und Siebertz [26] für das *model checking* von Formeln der Prädikatenlogik erster Stufe auf *nowhere dense* Klassen von Graphen. In der Praxis können akzeptable Laufzeiten erwartet werden für Graphen mit *bounded expansion* und insbesondere für Graphen, die K_t als Minor ausschließen für $t \in \mathbb{N}$. Der Algorithmus wurde in *C++* implementiert und es wird eine empirische Analyse der Laufzeit präsentiert. Für einige Graphen, die aus der Praxis stammen, terminiert der Algorithmus schnell, z.B. wird für ein Facebook ego-Netzwerk mit 2'888 Knoten und 2'981 Kanten ein *dominating set* der Größe 10 in weniger als drei Minuten gefunden. Auf strukturell komplexeren Graphen ist der Algorithmus aber sehr langsam. Es kann mehrere Tage dauern, ein *dominating set* der Größe 5 zu finden für einen Zufallsgraphen mit 30 Knoten, der mit hoher Wahrscheinlichkeit K_5 als Minor ausschließt.

Abstract

We present an algorithm to decide the dominating set problem (G, k) that is fixed parameter tractable on nowhere dense classes of graphs. The algorithm is based on the model-checking algorithm for first-order formulas on nowhere dense classes of graphs introduced by Grohe, Kreutzer, and Siebertz [26]. Acceptable running times in practice are expected for graphs with bounded expansion and in particular for graphs that exclude K_t as a minor for $t \in \mathbb{N}$. We have implemented the algorithm in *C++* and present an empirical analysis of its running time. For some graphs that arise in practice our algorithm terminates quickly, for instance it finds a dominating set of size 10 for a Facebook ego network graph with 2'888 vertices and 2'981 edges in less than three minutes. On structurally complex graphs, however, the algorithm is very slow. It may run several days to find a dominating set of size 5 for a random graph with 30 vertices that excludes K_5 as a minor with high probability.

Contents

List of Tables

1 Introduction

In the year 2014, Grohe, Kreutzer and Siebertz [26] developed a model-checking algorithm for first-order formulas on nowhere dense classes of graphs. They showed that for every nowhere dense class of graphs $\mathcal{C}$ there exists a function f such that for all first-order formulas φ, all real numbers $\varepsilon > 0$ and all graphs $G \in \mathcal{C}$ with n vertices, the problem whether G satisfies the formula φ, written $G \models \varphi$, can be decided in time $f(|\varphi|, \varepsilon) \cdot n^{\varepsilon}$. In other words, the first-order model-checking problem on nowhere dense classes of graphs is fixed-parameter tractable where we choose $|\varphi|$ as the parameter. Nowhere dense classes were introduced by Nešetřil and Ossona de Mendez [35] as a very general formalization of sparse graphs. These graph classes include all planar graphs, graphs with excluded minors or excluded topological minors. It was already known that the first-order model-checking problem is fixed-parameter tractable on all of these classes [23, 22, 21], so the result of Grohe, Kreutzer and Siebertz extends all earlier results. For classes of graphs which are closed under taking subgraphs this result is even the best that one can expect in the following sense. If the parameterized complexity class AW[$\star$] is not equal to the parameterized complexity class FPT, which is a standard assumption in parameterized complexity theory, and $\mathcal{C}$ is a class of graphs that is closed under subgraphs, then the first-order model-checking problem is fixed-parameter tractable on $\mathcal{C}$ if and only if $\mathcal{C}$ is nowhere dense.

An algorithmic theorem for the model-checking problem of a logic can be called an algorithmic meta-theorem because it provides a tractability result not only for a single problem but for the whole class of problems that are definable in first-order logic. A theorem like this is very useful because it provides a very easy way to prove that certain problems can be solved efficiently on certain classes of graphs. For example, we can easily prove that the dominating set problem is fixed-parameter tractable on any nowhere dense class of graphs simply by giving a first-order formula expressing the problem: there exists a dominating set of size at most k in a graph G if and only if the graph is a model of the following formula

$$\exists x_1 \dots \exists x_k \forall y \Big(\bigvee_{1 \leq i \leq k} (x_i = y \vee E(x_i, y)) \Big).$$

First-order logic is a very important logic and it can express many properties of graphs, such as generalized independent set problems, generalized dominating set problems and many more.

As stated above, the running time of the model-checking algorithm of [26] depends almost linearly on the size of the given input graph. Therefore, in theory, this model-checking algorithm can be applied to very large graphs if the property which we want to test can be expressed in a short first-order formula. For any practical implementation

however, the function f plays an important role, as it links computation time to the parameter $|\varphi|$. Let us give an example. Courcelle's theorem [13] states that every property of graphs definable in monadic-second order logic can be decided in linear time on any class of graphs of bounded tree-width. In theory, this implies that for a graph G of bounded treewidth and a short formula φ it can be decided quickly whether $G \models \varphi$. It was long believed however, that implementations of Courcelle's theorem could not have running times that are acceptable in practice. Courcelle's original proof used a translation from logical formulas to automata. For every negation in the formula, the automaton that is constructed inductively has to be complemented, which means, it has to be made deterministic. This is based on a powerset construction and hence yields an exponential blow-up of the automaton's size, and therefore a non-elementary growth of the function f. Despite the linear dependence on the graph size, this approach therefore renders model-checking practically infeasible when several negations occur in the formula. But recently, Courcelle's theorem has been implemented in a clever way by Langer, Reidl, Rossmanith, and Sikdar [32]. They avoided constructing the automata and implemented the model-checking problem with dynamic programming along the tree-decomposition of the graph. In a practical evaluation, their algorithm could compete e.g. with algorithms for the dominating set problem on graphs with bounded tree-width.

The aim of this thesis is to implement a fixed-parameter algorithm for the dominating set problem based on the approach of Grohe, Kreutzer and Siebertz [26]. At this point of time, we have little hope to obtain a feasible implementation for the general model-checking problem. The basis of the result of [26] is a translation of first-order formulas into a local normal form due to Gaifman [24]. As shown in [16], there is no elementary bound on the length of the local formula in terms of the input formula and we have no idea of how to avoid this in general. To make matters worse, the translation in [26] does not only have to be computed once, but recursively for several times. Furthermore, the general model-checking problem requires that we include neighborhood-covers into the logic, which makes a practical implementation even harder. We therefore believe that it is an interesting preliminary step to implement and evaluate the algorithm for several well behaved special cases. We implemented an algorithm for the dominating set problem, for which we can construct small local normal forms without referring to neighborhood covers. The dominating set problem is one of the most important first-order definable problems which has these desired features. Our implementation is optimized for classes of bounded expansion, an important subclass of nowhere dense classes of graphs. For these classes, a very recent result shows that the recursive search strategy of the algorithm terminates (theoretically) quickly [31].

We evaluated our algorithm to find minimum dominating sets on several graph instances that arise in practice and on several generated random graph instances with certain properties. Our results are encouraging on the one hand, but they also show that on structurally complex graphs, the algorithm is very slow. For example, we can compute a minimum dominating set of size 10 on a Facebook ego-network graph with 2'888 vertices and 2'981 edges in less than three minutes. On the other hand, the algorithm does not terminate within three weeks on several other graphs of similar size and larger stemming from social media, transportation, hyperlink networks and social

sciences. In the empirical analysis, each graph was handled in a single thread on an Intel Xeon E5-2697 (2.60 GHz) core on which no other tasks ran during our tests. Our evaluation of running times on random graphs that exclude K_5 as minor with high probability shows that even for such a graph with 30 vertices, the algorithm may run from several hours up to more than a week to find a dominating set. The size of the dominating sets that our algorithm found for these graphs ranges from five to six vertices. The extended running times are explained by surprisingly high values of the approximated weak coloring numbers of these graphs.

We present our work in the following chapters.

- In **Chapter 1**, we provide background from parameterized complexity theory and graph theory.
- In **Chapter 2**, we give a brief description of the model-checking algorithm of Grohe, Kreutzer and Siebertz [26].
- In **Chapter 3**, we present our implementation and prove its correctness.
- In **Chapter 4**, we present an analysis of running times of our algorithm on practical graph instances.
- We conclude in **Chapter 5**.

2 Background

2.1 Basics from graph theory

Our notation is standard and we refer to [18] for further background. All graphs in this thesis are finite, undirected and simple, i.e. neither loops nor multiple edges between the same pair of vertices exist. If G is a graph, then $V(G)$ denotes its set of *vertices* and $E(G)$ its set of *edges*.

Two vertices $v, w \in V(G)$ are *adjacent*, if there exists an edge between them, i.e. if $\{v, w\} \in E(G)$. The *neighborhood* of a vertex v is the set of adjacent vertices $N(v) = \{w \in V(G) \mid \{v, w\} \in E(G)\}$. By $N[v]$ we refer to the *closed neighborhood* of v, $N[v] = N(v) \cup \{v\}$. For a set of vertices $X \subseteq V(G)$, the neighborhood of X is the set of vertices that are adjacent to a vertex in X, excluding the vertices that are in X itself, formally $N(X) = \bigcup_{v \in X} N(v) \setminus X$. Similar to the case of a single vertex, we write $N[X]$ to refer to $\bigcup_{v \in X} N[v]$.

A graph G' is a *subgraph* of G, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. A subgraph G' of G is an *induced subgraph* of G if $E(G') = \{\{v, w\} \in E(G) \mid v, w \in V(G')\}$. For a set $X \subseteq V(G)$, we write $G[X]$ for the subgraph induced by X, that is, for the unique induced subgraph of G with vertex set X. If $v \in V(G)$, then $G - v$ refers to the subgraph induced by $V(G) \setminus \{v\}$.

A *path* P is a sequence of vertices $v_0 v_1 \dots v_k$ with $v_i \neq v_j$ for all $0 \leq i, j \leq k$ with $i \neq j$ and that is connected by the set of edges $\{\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}\}$. A path's length is given by the number of its edges. In a graph G, the *distance* between two vertices $v, w \in V(G)$, denoted by $d(v, w)$, equals the length of a shortest path connecting v and w. If there is no path in G that connects v and w, we define $d(v, w) := \infty$. A graph G with $V(G) \neq \emptyset$ is *connected*, if for each pair of vertices $v, w \in V(G)$, there exists a path P with $v, w \in P$, i.e. P is a path linking v and w. A *component* C of G is a maximally connected subgraph of G.

The *diameter* of a graph G, denoted by $\mathrm{diam}(G)$, equals the maximum distance between two vertices $v, w \in V(G)$. The *radius* of G, denoted $\mathrm{rad}(G)$, is defined as $min_{v \in V(G)} max_{w \in V(G)} d(v, w)$.

2.2 The dominating set problem

Let G be a graph. A set $D \subseteq V(G)$ is a *dominating set* if for every vertex $v \in V(G)$ either $v \in D$ or there exists $u \in D$ with $\{u, v\} \in E(G)$. In the latter case, v is *dominated* by u. D is a minimum dominating set if no other dominating set D' exists such that $|D'| < |D|$. The smallest value of k for which a dominating set D with $|D| = k$ exists is

called the *domination number* $\gamma(G)$ of G.

Definition 2.2.1. Dominating set problem
Input: *Graph G and $k \in \mathbb{N}$*
Problem: *Does G contain a dominating set of size at most k?*

For $r \in \mathbb{N}$, a set $D \subseteq V(G)$ is an *r-dominating set* if for every vertex $v \in V(G)$ there is a vertex $u \in D$ such that $d(u, v) \leq r$ (possibly $d(u, v) = 0$, when v itself lies in D).

Niedermeier refers to the dominating set problem as "a core problem in combinatorial optimization and graph theory" [10]. The importance of the dominating set problem is reflected in both the academic interest it has received and the number of its practical applications. As of 1998, already, there are more than 200 academic papers and 30 PhD theses examining domination problems and their complexity [28]. For a survey of practical applications, see [27, 39].

2.3 Parameterized complexity

In this section, we quickly review the basic definitions from parameterized complexity theory. We refer to [37] for more background.

In the field of parametrized complexity, the running time of algorithms is analyzed not only depending on the size of the input, but as a function of both the size of the input and an additional parameter.

Definition 2.3.1. (Niedermeier [37]) *A* parameterized problem *is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a finite alphabet. The second component is called the parameter of the problem.*

Definition 2.3.2. (Niedermeier [37]) *A parameterized problem L is* fixed-parameter tractable *if it can be determined in $f(k) \cdot n^{O(1)}$ time whether or not $(x, k) \in L$, where f is a computable function. The parameterized complexity class containing all fixed-parameter tractable problems is called FPT.*

As Niedermeier [37] points out, FPT algorithms generally solve NP-hard problems and, as a result, their running time may grow exponentially with the size of the parameter k. Nevertheless, from a practical point of view, FPT algorithms shall *efficiently* solve problems, that is they are supposed to solve real world problems in reasonable time. This can be achieved by keeping the growth in k modest or by finding a parameterization of the problem such that k takes only relatively small values.

This indicates two perspectives on fixed parameter tractability. On the one side, there is the theoretical point of view, that asks whether or not a given parameterized problem L is in FPT. From a practical point of view, on the other side, the focus is on the running time an FPT algorithm requires to decide real world instances (x, k) of the problem L. Regarding the practical perspective on FPT, it should be stressed that FPT analyzes worst case complexity. Despite a possibly expensive growth of the running time in k, a FPT algorithm may relatively efficiently solve real world instances (x, k) if it successfully exploits their structure [37].

In general, the dominating set problem is known to be NP-complete [25]. With respect to fixed parameter tractability, we are working with the following parameterization of the dominating set problem. The first component of the language $L \subseteq \Sigma^* \times \Sigma^*$ is a graph G, encoded over the alphabet Σ. The parameter of the problem, that is the second component of L, is k, the maximum number of vertices allowed in a dominating set. Since $k \in \mathbb{N}$, in the following we assume that $L \subseteq \Sigma^* \times \mathbb{N}$. If a dominating set of size at most k exists for graph G, then $(G, k) \in L$. In general, the problem is known to be W[2]-complete with respect to this parameterization [19]. As a result, the existence of an FPT algorithm solving the dominating set problem on arbitrary classes of graphs is very unlikely [10]. However, if more is known about the structure of a graph, this structure can be exploited algorithmically, which leads to more efficient algorithms.

2.4 Kernelization

Kernelization is a technique for data reduction of parameterized problems by preprocessing. Its aim is to simplify a given problem instance by reducing its size.

Definition 2.4.1. (Niedermeier [37]) *Let L be a parameterized problem, that is, L consists of input pairs (x, k), where x is the problem instance and k is the parameter.* Reduction to a problem kernel *then means to replace (x, k) by a "reduced" instance (x', k') (called* problem kernel*) such that*

$$k' \leq k, \qquad |x'| \leq g(k)$$

for some function g only depending on k, and

$$(x, k) \in L \text{ iff } (x', k') \in L.$$

Furthermore, the reduction from (x, k) to (x', k') must be computable in polynomial time $T_K(|x|, k)$. The function $g(k)$ is called the size of the kernel.

For the empirical analysis of our algorithm we implemented two polynomial time data reduction rules introduced by Alber, Fellows, and Niedermeier [10]. Given a graph G and a vertex $v \in V(G)$, these rules are based on the division of $N(v)$ into the sets

$$\begin{aligned} N_1(v) &:= \{u \in N(v) \mid N(u) \setminus N[v] \neq \emptyset\}, \\ N_2(v) &:= \{u \in N(v) \setminus N_1(v) \mid N(u) \cap N_1(v) \neq \emptyset\} \text{ and} \\ N_3(v) &:= N(v) \setminus (N_1(v) \cup N_2(v)). \end{aligned}$$

Rule 1. (Alber, Fellows, and Niedermeier [10]) *If $N_3(v) \neq \emptyset$ for some vertex v, then*

- *remove $N_2(v)$ and $N_3(v)$ from G and*
- *add a new vertex v' with the edge $\{v, v'\}$ to G.*

Rule 2 considers the neighborhood of two vertices $v, w \in V(G)$. Let $N(v,w) \coloneqq N(v) \cup N(w) \setminus \{v,w\}$ and $N[v,w] \coloneqq N[v] \cup N[w]$. The set $N(v,w)$ is partitioned into

$$N_1(v,w) \coloneqq \{u \in N(v,w) \mid N(u) \setminus N[v,w] \neq \emptyset\},$$
$$N_2(v,w) \coloneqq \{u \in N(v,w) \setminus N_1(v,w) \mid N(u) \cap N_1(v,w) \neq \emptyset\} \text{ and}$$
$$N_3(v,w) \coloneqq N(v,w) \setminus (N_1(v,w) \cup N_2(v,w)).$$

Rule 2. (Alber, Fellows, and Niedermeier [10]) *Consider $v, w \in V(G)$ $(v \neq w)$ and suppose that $|N_3(v,w)| > 1$. Suppose that $N_3(v,w)$ cannot be dominated by a single vertex from $N_2(v,w) \cup N_3(v,w)$.*

Case 1 *If $N_3(v,w)$ can be dominated by a single vertex from $\{v,w\}$:*

1.1 *If $N_3(v,w) \subseteq N(v)$ as well as $N_3(v,w) \subseteq N(w)$:*
- *remove $N_3(v,w)$ and $N_2(v,w) \cap N(v) \cap N(w)$ from G and*
- *add two new vertices z, z' and edges $\{v,z\}$, $\{w,z\}$, $\{v,z'\}$, $\{w,z'\}$ to G.*

1.2 *If $N_3(v,w) \subseteq N(v)$, but not $N_3(v,w) \subseteq N(w)$:*
- *remove $N_3(v,w)$ and $N_2(v,w) \cap N(v)$ from G and*
- *add a new vertex v' and the edge $\{v,v'\}$ to G.*

1.3 *If $N_3(v,w) \subseteq N(w)$, but not $N_3(v,w) \subseteq N(v)$:*
- *remove $N_3(v,w)$ and $N_2(v,w) \cap N(w)$ from G and*
- *add a new vertex w' and the edge $\{w,w'\}$ to G.*

Case 2 *If $N_3(v,w)$ cannot be dominated by a single vertex from $\{v,w\}$*
- *remove $N_3(v,w)$ and $N_2(v,w)$ from G and*
- *add a two new vertices v', w' and edges $\{v,v'\}, \{w,w'\}$ to G.*

Alber, Fellows, and Niedermeier [10] show that Rule 1 can be carried out in $O(n)$ for planar graphs and $O(n^3)$ for general graphs. For Rule 2 the corresponding complexities are $O(n^2)$ and $O(n^4)$.

Any vertex added according to these reduction rules can be considered a gadget vertex. The vertex v' and the edge $\{v,v'\}$ added in Rule 1, for instance, enforce that either v or v' will be included in the dominating set. For the empirical analysis, instead of adding a gadget vertex, we immediately add a vertex to which any reduction rule (except Rule 2 Case 1.1) can be applied to the dominating set. This approach is also taken by Alber, Fellows, and Niedermeier [10] in their empirical analysis. More details will be given in Section 4.1.

2.5 Structural graph theory

2.5.1 Degenerate graphs

A graph G is k-degenerate if every subgraph $H \subseteq G$ of G has a vertex of degree at most k. A graph class is called degenerate if there exists a number k such that every graph $G \in \mathcal{C}$ is k-degenerate. Classically, degenerate graphs are considered as uniformly sparse graphs. It has been shown that the dominating set problem is fixed-parameter tractable on degenerate graphs [38]. However, this is not true for the more general r-dominating set problem. For example the 2-dominating set problem on the class of all graphs where every edge is subdivided exactly once trivially reduces to the dominating set problem on the class of all graphs. This class however is generate. The same trick can be used to show that the r-dominating set problem is W[2]-hard on every somewhere dense class of graphs that is closed under subgraphs [21]. We will define somewhere dense and nowhere dense classes of graphs below.

2.5.2 Graphs with excluded minors

An important degenerate class of graphs is the class of all planar graphs or more generally, any class of graphs that excludes a fixed minor. Let G and H be graphs. If H can be obtained from a subgraph of G by contracting edges, then H is a *minor* of G. This relationship is denoted by $H \preceq G$. Equivalently, minors can be defined using *minor models*. G has a model of H, if for every $v \in V(H)$, there exists a tree $T_v \subseteq G$ with T_u and T_v disjoint if $u \neq v$. Furthermore, for all edges $\{u, v\} \in E(H)$, it is required that there is an edge in G between an arbitrary vertex in T_u and an arbitrary vertex in T_v. The trees T_v are referred to as the *branch sets* of the minor model. A class $\mathcal{C}$ of graphs excludes a minor if there is some graph H such that $H \not\preceq G$ for all $G \in \mathcal{C}$. Such classes are sometimes simply called H-minor free classes, even if no minor H is specified. If a complete graph K_t is excluded from G, then G has edge density bounded by $(\alpha + o(1)) \cdot t\sqrt{\ln t}$, where $\alpha = 0.319$ is a constant [40]. It is also shown in [40] that the graphs which realize this extremal edge density are random graphs with this density. We are going to test our dominating set algorithm (among others) on such random graphs, because these results provide us with an upper bound on the expected running time on these graphs.

It is known that the dominating set problem is fixed-parameter tractable on planar graphs [9] and on classes that exclude a fixed minor using bidimensionality [17].

2.5.3 Nowhere dense classes of graphs

Let G be a graph and let $r \in \mathbb{N}$. A *depth-r minor* of G is a graph H, such that $H \preceq G$, which is witnessed by collection of branch sets $\{T_v \mid v \in V(H)\}$ such that $\mathrm{rad}(T_v) \leq r$. If H is a depth-r minor of G, we write $H \preceq_r G$.

Nešetřil and Ossona de Mendez [35] introduced nowhere dense classes of graphs as a model of uniformly sparse graphs. Originally, the definition of nowhere dense classes of graphs was in terms of densities of bounded depth minors. For our purpose, it is more

convenient to work with an equivalent definition in terms of excluded bounded depth minors and in terms of the splitter game.

Definition 2.5.1. *A class $\mathcal{C}$ of graphs is* nowhere dense *if for every positive integer r, there is a graph H_r such that $H_r \not\preceq_r G$ for all $G \in \mathcal{C}$. $\mathcal{C}$ is* effectively nowhere dense *if there is a computable function f such that for every positive integer r the complete graph $K_{f(r)}$ is excluded as a depth-r minor from all graphs $G \in \mathcal{C}$.*

An important note is that the class of all 2-degenerate graphs is not nowhere dense, hence the classical notion of sparsity is not generalized by nowhere dense classes of graphs. The fact that the r-dominating set problem is W[2]-hard on 2-degenerate graphs shows however, that the classical notion of sparseness is not the right notion to consider in the model-checking context.

Grohe, Kreutzer, and Siebertz [26] introduced the following splitter game, which leads to a game theoretic characterization of nowhere dense classes. We use the simple splitter game, which is easier to implement and which has some nice features, as we will see later.

Let G be a graph and let $\ell, r > 0$. The *simple ℓ-round radius-r splitter game* on G is played by two players, Connector and Splitter, as follows. We let $G_0 := G$. In round $i+1$ of the game, Connector chooses a vertex $v_{i+1} \in V(G_i)$. Then Splitter picks a vertex $w_{i+1} \in N_r^{G_i}(v_{i+i})$. We let $G_{i+1} := G_i[N_r^{G_i}(v_{i+1}) \setminus \{w_i\}]$. Splitter wins if $G_{i+1} = \emptyset$. Otherwise the game continues at G_{i+1}. If Splitter has not won after ℓ rounds, then Connector wins.

A *strategy* for Splitter is a function that maps every partial play $(v_1, w_1, \ldots, v_s, w_s)$ with associated sequence $G_0, \ldots, G_s$ of graphs and move $v_{s+1} \in V(G_s)$ by Connector to a vertex $w_{s+1} \in N_r^{G_s}(v_{s+1})$. A strategy is a *winning strategy* for Splitter in the simple ℓ-round radius-r splitter game on G if Splitter wins every play in which he follows the strategy f. If Splitter has a winning strategy, we say that he *wins* the simple ℓ-round radius-r splitter game on G.

Theorem 2.5.1. (Grohe, Kreutzer, and Siebertz [26]) *Let $\mathcal{C}$ be a nowhere dense class of graphs. Then for every positive integer r there is an integer ℓ such that for every $G \in \mathcal{C}$, Splitter wins the simple ℓ-round radius-r splitter game on G. Furthermore, for every partial play $(v_1, w_1, \ldots, v_s, w_s)$ with associated sequence $G_0, \ldots, G_s$ of graphs and move $v_{s+1} \in V(G_s)$ by Connector, Splitter's answer w_{s+1} can be computed efficiently.*

The splitter game provides us with a powerful technique of recursive graph decompositions. In particular, for the parameterized dominating set problem, we will construct a search tree whose degree is bounded by a function of the size of the solution and whose depth is bounded by the number of rounds that Splitter needs to win the game.

For every nowhere dense class of graphs there is a function ℓ such that for every integer r Splitter wins the simple $\ell(r)$-round radius-r splitter game. Unfortunately, the function ℓ for a general nowhere dense class is huge. The proof in [26] showing the existence of ℓ is based on a characterisation of nowhere dense classes in terms of uniformly quasi-wideness. We are not going to repeat the definition here because it does not help the understanding of our results, however, we state that the bounds that can

be derived from uniformly quasi-wideness form a tower of exponentials of height r. The best known bounds are not even improved for several well understood nowhere dense classes of graphs, e.g. in [11] it is shown that every class that excludes a fixed minor is uniformly quasi-wide and the corresponding upper bounds are not smaller than in the general case. Only recently [31], new bounds for the length ℓ of the splitter game were given in terms of the generalized colouring numbers.

2.5.4 Classes of bounded expansion and the generalized coloring numbers

We have defined effectively nowhere dense classes by the existence of a function f such that $K_{f(r)}$ is excluded as a depth-r minor of every graph from the class. This means that a class $\mathcal{C}$ of graphs is nowhere dense if for every integer r, the clique number $\omega(H)$ is bounded by a function $f(r)$ for every depth-r minor H of any graph $G \in \mathcal{C}$. If we slightly change this definition, we obtain classes of bounded expansion. For a graph G, $\chi(G)$ denotes the chromatic number of G.

Definition 2.5.2. *A class $\mathcal{C}$ of graphs has* bounded expansion *if there is a function f such that for every positive integer r, every graph $H \preceq_r G$ for $G \in \mathcal{C}$ satisfies $\chi(H) \leq f(r)$.*

We are going to work with an equivalent definition in terms of the generalized coloring numbers. The generalized coloring numbers have been introduced by Kierstead and Yang [30]. The name general coloring numbers suggests that there is a direct connection to the chromatic number. This connection is in the following sense. The generalised 1-coloring number is equal to the degeneracy of the graph plus one and the degeneracy plus one of a graph is an upper bound for the chromatic number of a graph.

Let G be a graph and let $r \in \mathbb{N}$. By $\Pi(G)$ we refer to the set of linear orders on $V(G)$. Given an order $\leq \in \Pi(G)$, a vertex $u \in V(G)$ is weakly r-reachable from $v \in V(G)$ if the following holds. Between u and v there exists a path P of length $\ell \leq r$, such that each $u' \in V(P)$ satisfies $u \leq u'$. That is, u is the minimum of $V(P)$ with respect to the order $\leq$. For $v \in V(G)$ and $\leq \in \Pi(G)$, we write $\mathrm{WReach}_r[G, \leq, v]$ to refer to set of vertices which are weakly r-reachable from v with respect to $\leq$.

We can now move on to the definitions of the coloring numbers. By $\mathrm{wcol}_r(G)$ we refer to the weak r-coloring number of graph G. It is defined as:

$$\mathrm{wcol}_r(G) = \min_{\leq \in \Pi(G)} \max_{v \in V(G)} |\mathrm{WReach}_r[G, \leq, v]|.$$

Theorem 2.5.2. (Zhu [42]) *A class $\mathcal{C}$ of graphs has bounded expansion if and only if there is a function f such that $\mathrm{wcol}_r(G) \leq f(r)$ for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$.*

Our motivation to study the generalized coloring numbers comes from the following theorem.

Theorem 2.5.3. (Kreutzer, Pilipczuk, Rabinovich, and Siebertz [31]) *Let G be a graph, let $r \in \mathbb{N}$ and let $\mathrm{wcol}_{2r}(G) \leq \ell$. Then Splitter wins the simple ℓ-round radius-r splitter game. Furthermore, if an order $\leq$ that witnesses $\mathrm{wcol}_{2r}(G) \leq \ell$ is given and if*

$(v_1, w_1, \ldots, v_s, w_s)$ is a partial play with associated sequence $G_1, \ldots, G_s$ of graphs, then on Connector's choice v_{s+1}, it is a winning strategy for Splitter to choose w_{s+1} as the minimal element of $N_r^{G_s}(v_{s+1})$ with respect to $\leq$.

Our algorithm will be based on this very simple strategy described in the above lemma. For every class of graphs on which we know how to efficiently compute high quality orderings, we have good hope that our algorithm will terminate quickly. In particular, on classes that exclude a fixed minor K_t, we know that if $K_t \not\preceq G$, then $\mathrm{wcol}_r(G) \in O(r^t)$ and a good order can be efficiently computed [41]. For planar graphs, even better approximations are known, however, we have not implemented these.

Another motivation to study the generalized coloring numbers comes from the fact that they allow the construction of sparse neighborhood covers. Let G be a graph and let $r \in \mathbb{N}$. An r-neighborhood cover $\mathcal{X}$ of G is a set of connected subgraphs of G such that for all $v \in V(G)$ there is some $X \in \mathcal{X}$ with $N_r(v) \subseteq X$. The radius of $\mathcal{X}$ is defined as $\max\{\mathrm{rad}(X) : X \in \mathcal{X}\}$. The degree of $\mathcal{X}$ is defined as $\max |\{X \in \mathcal{X} : v \in V(X)\}|$.

Theorem 2.5.4. (Grohe, Kreutzer, and Siebertz [26]) *Let G be a graph and let $r \in \mathbb{N}$. Then G admits an r-neighborhood cover of radius at most $2r$ and degree at most $\mathrm{wcol}_{2r}(G)$.*

This implies that sparse neighborhood covers of small radius can even be computed for nowhere dense classes of graphs. However, their computation is in general not feasible for large graphs for larger values of r (it is based on the computation of transitive fraternal augmentations where the dependence on r is a polynomial in the densities of depth-r minors of degree about 2^{2^r}). We refer to Chapter 7.4 of [36] for a discussion on this matter. This is another motivation to study classes of graphs for which we know how to compute weak coloring numbers and sparse neighborhood covers, accordingly.

It was shown that the dominating set problem and more generally, the r-dominating set problem is fixed-parameter tractable on any nowhere dense class of graphs [15]. Very recently, it was shown that the dominating set problem admits an almost linear kernel on these graphs classes [20]. It is open whether polynomial kernels for the general r-dominating set problem exist.

3 First-order model-checking and dominating sets on nowhere dense classes of graphs

A recent result shows that every first-order definable property of nowhere dense classes of graphs can be decided efficiently [26]. We are going to follow this very general approach based on the splitter game to solve the dominating set problem. Our aim is not necessarily to implement the most efficient algorithm for the dominating set problem but rather to test whether the general method presented in [26] has the potential to be implemented for much more general graph properties. In particular, we want to test whether splitter game based methods produce recursive search trees of acceptable depths. However, we have chosen a first-order definable problem that does not require the implementation of the whole approach presented in [26]. In particular, it is not necessary to base the data structures in the search tree on neighbourhood covers. As we will present in the next section, we can work with components of small radius instead.

3.1 Gaifman's theorem

We give only a very brief background from logic. For more background we refer to [33]. We fix a finite relational vocabulary τ. A finite τ-structure A consists of a finite set $V(A)$, called the universe of A and a relation of the right arity for each relation symbol from τ. For example, a graph can be understood as a $\{E\}$-structure, where E is a binary relation symbol. First-order formulas of vocabulary τ are formed from atomic formulas $x = y$ and $R(x_1, \ldots, x_k)$, where $R \in \tau$ is a k-ary relation symbol and $x, y, x_1, \ldots, x_k$ are variables and from the usual Boolean connectives $\neg$, $\vee$, $\wedge$ and existential and universal quantification $\exists x$, $\forall x$, respectively. A sentence is a formula without free variables. The quantifier rank of a formula is the nesting depth of quantifiers in the formula.

The starting point of the result in [26] is Gaifman's theorem, which states that every first-order sentence φ is equivalent to a Boolean combination of basic-local sentences, that is, of sentences of the form

$$\exists x_1 \ldots \exists x_k \big(\bigwedge_{i \neq j} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_i \psi(x_i) \big),$$

where $\psi(x)$ is a formula that is local, that is, whose truth value in a structure depends only on the r-neighborhood of x_i. The numbers k and r depend only on φ and furthermore, this Gaifman normal form can be computed effectively from φ.

The question whether a graph G has a dominating set of size k can be formulated in first-order logic as follows. It holds that G has such a dominating set if and only if it satisfies the sentence ψ_k:

$$\psi_k = \exists x_1 \dots \exists x_k \forall y (\bigvee_{1 \leq i \leq k} (x_i = y \vee E(x_i, y)).$$

It is easy to find a Gaifman normal form for this sentence by hand. For this, observe that every component of G that contains j vertices from a dominating set can have radius at most $3j$. Furthermore, observe that the quantifier-rank of the formula in Gaifman normal form does not increase. In general, this is not the case. In fact, it was shown in [16] that the quantifier-rank may grow non-elementarily, when translating a formula into Gaifman normal form.

With a formula in Gaifman normal form at hand, one has to compute for the local neighborhoods of all elements whether they satisfy the local formula ψ. This is done recursively in the algorithm of [26]. Based on the splitter game, a single vertex is deleted from the neighborhood of each vertex. A first-order formula can deal with the deletion of a single vertex (or any fixed number of vertices) if its neighbors are colored. This allows the formula to recover the original formula from the colored graph. Because the splitter game terminates after a bounded number of rounds, this can be used to check whether the formula holds. In each step, after deleting the vertex and rewriting the formula to deal with this deletion, one has to recompute the Gaifman normal form. It was a major part of [26] to provide an extended Gaifman normal form whose quantifier rank remains stable under such deletions and re-computations of Gaifman normal forms. Furthermore, it is too expensive to compute the local types of every element individually. Instead, based on neighborhood covers, local evaluations for many elements are grouped in clusters and solved simultaneously. This makes matters even more complicated, as neighborhood covers have to be integrated into the logic and the locality theorem must deal also with the covers.

In the following section, we show that the dominating set problem can be generalised to a colored version, such that it is stable under deletions of vertices just as in the general case. Furthermore, we will show that it is not necessary to incorporate neighborhood covers into the problem. For the original dominating set problem it is easy to see that each component has small radius. We show that the colored variant preserves this property if we enrich the problem with an additional hitting set problem. This allows us to use the trivial neighborhood cover consisting of one cluster for each component, which can simply be dropped from the formulation.

3.2 Generalized dominating sets

We introduce two variants of the dominating set problem that allow our algorithm to encode information in colors. The information required to be encoded concerns vertices that are already dominated and vertex sets of which at least one vertex of each set must

be added to a dominating set. Note that the occurring colors represent a very simplified version of types in the general first-order model-checking approach presented in [26].

Let G be a graph, $H, F_1, \ldots, F_l \subseteq V(G)$ colors for $l \in \mathbb{N}$ and $\mathcal{F} \coloneqq \{F_1, \ldots, F_l\}$. We refer to the set of all colors by $\mathcal{M} \coloneqq \{H, F_1, \ldots F_l\}$. A vertex v is colored $M \in \mathcal{M}$ if $v \in M$ or uncolored if $v \notin \bigcup_{M \in \mathcal{M}} M$. The colors of a vertex v are $\{M \in \mathcal{M} \mid v \in M\}$.

A generalized dominating set D' is a dominating set of $G - H$ which satisfies the following constraint related to $\mathcal{M}$. For each $F' \in \mathcal{F}' \coloneqq \{F \in \mathcal{F} \mid F \neq \emptyset\}$ at least one vertex v colored F' is contained in D'. That is $F' \cap D' \neq \emptyset$ for all $F' \in \mathcal{F}'$. We will refer to this as hitting each color $F' \in \mathcal{F}'$. Note that D' may contain vertices of H.

Definition 3.2.1. Generalized dominating set problem
Input: *Graph G, colors $H, F_1, \ldots, F_l \subseteq V(G)$ and $k \in \mathbb{N}$*
Problem: *Does G contain a generalized dominating set of size at most k?*

The restricted generalized dominating set problem is given by the general dominating set problem with the additional restriction that colors $H, F_1, \ldots, F_l$ are a subset of $N[V(G) \setminus H]$:

Definition 3.2.2. Restricted generalized dominating set problem
Input: *Graph G, colors $H, F_1, \ldots, F_l \subseteq V(G)$ with $H, F_1, \ldots, F_l \subseteq N[V(G) \setminus H]$ and $k \in \mathbb{N}$*
Problem: *Does G contain a generalized dominating set of size at most k?*

We will show that it suffices to consider only the restricted version of the problem.

3.3 The `RGDS` algorithm

We are now going to introduce the algorithm `RGDS` that decides the restricted generalized dominating set problem for the input graph G, colors $H, F_1, \ldots, F_l \subseteq V(G)$, $k \in \mathbb{N}$ and a set of vertices D. Remember that we write $\mathcal{F}$ to refer to the set of colors $\{F_1, \ldots, F_l\}$. If $(G, H, \mathcal{F}, k)$ is an instance of the generalized dominating set problem, `RGDS` returns a generalized dominating set D' with $|D'| \leq k$. Otherwise, the return value is `Null` to signalize that $(G, H, \mathcal{F}, k)$ is not an instance of the generalized dominating set problem.

Regarding the distinction between dominating set, generalized dominating set and restricted generalized dominating set, we note the following. The motivation for the development of `RGDS` is, actually, to decide the dominating set problem (G, k).

Lemma 3.3.1. *Let G be a graph and $k \in \mathbb{N}$. $(G, \emptyset, \emptyset, k)$ is a solution of the generalized dominating set problem if and only if (G, k) is a solution of the dominating set problem.*

Proof. In the context of generalized dominating set, $H = \emptyset$ means that no vertex is marked as already dominated, so each vertex $v \in V(G)$ has to be dominated by a generalized dominating set. Furthermore, no requirements concerning vertices that must be included in a generalized dominating set are made if $\mathcal{F} = \emptyset$. It follows that a set D which is a witness that $(G, \emptyset, \emptyset, k)$ is a solution of the generalized dominating set problem is also a witness that (G, k) is a solution of the dominating set problem. Conversely,

for the same reason, a dominating set D for (G, k) is a generalized dominating set for $(G, \emptyset, \emptyset, k)$. □

Hence, for the initial call of `RGDS` (see Algorithm 1 below) it must hold that $H = \emptyset$ and $\mathcal{F} = \emptyset$. With respect to parameter D, we also demand $D = \emptyset$ for the initial call of `RGDS`. In the algorithm D serves as a container to pass vertices added to a generalized dominating set along to recursive calls of `RGDS`. Passing D as a parameter allows `RGDS` to return a generalized dominating set.

By passing empty colors only to the initial call of `RGDS`, we clearly meet the algorithm's requirement that $H, F_1, \ldots, F_l \subseteq N[V(G) \setminus H]$. We will show below that the same holds for all recursive calls of `RGDS`. After the description of the algorithm, the proof of its correctness is given in Section 3.4.

The algorithm is based on the idea that for any vertex $v \in V(G)$ to be dominated, we either have to add v itself or $u \in N(v)$ to D'. `RGDS` tries both possibilities, that is we have one branch for $v \in D'$ and one for $v \notin D'$. In either branch, we delete v and color its neighborhood such that we store all required information related to the decision $v \in D'$ respectively $v \notin D'$.

Algorithm 1 Restricted generalized dominating set

Require: Graph G, colors $H, F_1, \ldots, F_l \subseteq V(G)$, $\mathcal{F} = \{F_1, \ldots, F_l\}$, $k \in \mathbb{N}$, set of vertices D
Require: $H, F_1, \ldots, F_l \subseteq N[V(G) \setminus H]$
Ensure: **return** Gen. dom. set D' with $|D'| \leq k$ if G contains such a D', else `Null`

1: **procedure** RGDS$(G, H, \mathcal{F}, k, D)$
2: **if** $k == 0$ **then return** $(V(G) \setminus H) \cup \bigcup_{F \in \mathcal{F}} F == \emptyset \ ? \ D \ : \ \texttt{Null}$ **end if**
3: **if** $|V(G)| \leq 1$ **then return** $(V(G) \not\subseteq H \vee \mathcal{F} \neq \emptyset) \ ? \ D \cup V(G) \ : \ D$ **end if**
4: $v \leftarrow$ choose $\in V(G)$

5: *// try $v \in$ GDS*
6: $W \leftarrow V(G - v) \setminus (H \cup N(v))$
7: $\mathcal{C} \leftarrow \{C \mid C \text{ component of } G[N[W]]\}$
8: $D' \leftarrow \text{SOLVE}(\mathcal{C}, H \cup N(v), \mathcal{F} \setminus \{F \in \mathcal{F} \mid v \in F\}, k - 1, D, V(G - v))$
9: **if** $D' \neq \texttt{Null}$ **then return** $D' \cup \{v\}$ **end if**

10: *// try $v \notin$ GDS*
11: **if** $N(v) = \emptyset \wedge v \notin H$ **then return** `Null` **end if**
12: $W \leftarrow V(G - v) \setminus H$
13: $\mathcal{C} \leftarrow \{C \mid C \text{ component of } G[N[W]]\}$
14: **if** $N(v) \neq \emptyset \wedge v \notin H$ **then** $\mathcal{F} \leftarrow \mathcal{F} \cup \{N(v)\}$ **end if**
15: **return** SOLVE$(\mathcal{C}, H, \mathcal{F}, k, D, V(G - v))$
16: **end procedure**

Claim 3.1. *The algorithm* `RGDS` *terminates on input* $(G, H, \mathcal{F}, k, D)$.

Proof. In each call of `RGDS` a vertex $v \in V(G)$ is the deleted and `RGDS` is called recursively on $G - v$ respectively on supgraphs of $G - v$. One of the termination conditions is $|V(G)| \leq 1$. Therefore, the algorithm is going to terminate. □

If we add v to D', the neighbors of v become dominated and to store this information, we add $N(v)$ to H, that is we define $H' \coloneqq H \cup N(v)$. The set $\mathcal{F}^- \coloneqq \{F \in \mathcal{F} \mid v \in F\}$ is the set of colors F which are hit by adding v to D'. These colors can be removed from $\mathcal{F}$, so we set $\mathcal{F}' \coloneqq \mathcal{F} \setminus \mathcal{F}^-$. Since we have added a vertex to D, k must be decremented by 1 and we get $k' \coloneqq k - 1$.

For the purpose of finding a generalized dominating set, this leaves us done with v (in case $v \in D'$, at least, the case $v \notin D'$ is handled below). The information related to v has been stored by adding $N(v)$ to H and removing F colors hit by v from $\mathcal{F}$. Hence, we can remove v from G and refer to the remaining graph as $G' \coloneqq G - v$.

Claim 3.2. *Let $W \coloneqq V(G') \setminus H$. After deleting v from G, the subgraphs to be passed on to recursive calls of* `RGDS` *are the components $\mathcal{C} \coloneqq \{C \mid C$ component of $G'[N[W]]\}$.*

Proof. In the context of generalized dominating set, any vertex $w \in W$ remains to be dominated, since it is not colored H. To dominate a vertex $w \in W$, it might be optimal to add vertex $v \notin W$ with $\{v, w\} \in E(G')$ to a generalized dominating set D'. It follows that the subgraphs of G' to be passed on to recursive calls of `RGDS` are the components of $G'[N[W]]$. □

We refer to the vertices not contained in any component $C \in \mathcal{C}$ as backland $B \coloneqq V(G') \setminus \bigcup_{C \in \mathcal{C}} V(C)$. The vertices in B are already dominated, otherwise they would be included in a component in $\mathcal{C}$. Though, we cannot discard B due to the requirement that any color $F \in \mathcal{F}$ must be hit, that is $F \cap D' \neq \emptyset$ for all $F \in \mathcal{F}$. Any vertex $b \in B$ still might be needed for that purpose.

Now, we are looking for a set of vertices $D'' \subseteq V(G')$ with $|D''| \leq k'$ such that all vertices in W are dominated and all colors in $\mathcal{F}'$ hit by D''. Such a D'' is a generalized dominating set for $(G', H', \mathcal{F}', k')$. We handle the task of looking for D'' in a separate algorithm, `Solve` (see Algorithm 2 below), since it will be repeated in the branch $v \notin D'$. `Solve` receives the input $(\mathcal{C}, H', \mathcal{F}', k', D \cup \{v\}, V(G'))$[1]. Before actually trying to find a generalized dominating set, we can check for three termination conditions.

Claim 3.3. *If either of the following conditions holds,* `Solve` *can immediately terminate and return* `Null`*.*
1) $|\mathcal{C}| > k'$.
2) It exists $C \in \mathcal{C}$ with diam $C > 3k' + 1$.
3) $\sum_{C \in \mathcal{C}}$ diam $C > 3k' + |\mathcal{C}|$.

Proof. The first termination condition follows from the definition of $\mathcal{C}$. For each $C \in \mathcal{C}$ it holds that $V(C) \cap W \neq \emptyset$ and hence at least one vertex is required to dominate C. If $|\mathcal{C}| > k'$, we cannot find a generalized dominating set for $(G', H', \mathcal{F}', k')$. The second

[1] $V(G')$ is required to calculate B, which is, in the pseudocode, calculated in `Solve`.

Algorithm 2 Verify if GDS has solution for components $\mathcal{C}$ and $H, \mathcal{F}, k$

Require: Components $\mathcal{C}$, $H, F_1, \ldots, F_l \subseteq V(G)$, $\mathcal{F} = \{F_1, \ldots, F_l\}$, $k \in \mathbb{N}$

procedure SOLVE($\mathcal{C}, H, \mathcal{F}, k, D, V(G)$)
  if $|\mathcal{C}| > k$ **then return** `Null` **end if**
  if $\exists C \in \mathcal{C}$. diam $C > 3k + 1$ **then return** `Null` **end if**
  if $\sum_{C \in \mathcal{C}}$ diam $C > 3k + |\mathcal{C}|$ **then return** `Null` **end if**
  $B \leftarrow V(G) \setminus \bigcup_{C \in \mathcal{C}} V(C)$
  for all $f \in \{f : \mathcal{F} \to \mathcal{C} \cup \{B\} \mid f \text{ function }\}$ **do**
    for all $C \in \mathcal{C}$ **do**
      if $\exists F \in f^{-1}(C)$. $F \neq \emptyset \wedge F \cap V(C) = \emptyset$ **then** try next f **end if**
      $\mathcal{F}' \leftarrow \{F_i \cap V(C) \mid F_i \in f^{-1}(C)\}$
      $H' \leftarrow H \cap V(C)$
      $k_{f^{-1}(C),C} \leftarrow \min\{k' \leq k \mid \text{RGDS}(C, H', \mathcal{F}', k', D) \neq \texttt{Null}\}$
      $D_C \leftarrow$ return value of RGDS$(C, H', \mathcal{F}', k_{f^{-1}(C),C}, D)$
    end for
    if $\exists F \in f^{-1}(B)$. $F \neq \emptyset \wedge F \cap B = \emptyset$ **then** try next f **end if**
    $(H_B, k_{f^{-1}(B),B}) \leftarrow$ MINHS$(\{F_i \cap B \mid F_i \in f^{-1}(B)\})$
    if $(\sum_{C \in \mathcal{C}} k_{f^{-1}(C),C}) + k_{f^{-1}(B),B} \leq k$ **then**
      return $D \cup \bigcup_{C \in \mathcal{C}} D_C \cup H_B$
    end if
  end for
  return `Null`
end procedure

termination condition follows immediately from Lemma 3.3.2. If the diameter of any component may not exceed $3k' + 1$, all components in $\mathcal{C}$ can only be dominated if $\sum_{C \in \mathcal{C}}$ diam $C > 3k' + |\mathcal{C}|$, which is the third termination condition. □

Lemma 3.3.2. *Given a component $C \in \mathcal{C}$ and $k' \in \mathbb{N}$, C does not contain a generalized dominating set of size at most k' if diam $C > 3k' + 1$.*

Proof. Let $C' = C[V(C) \setminus H]$ be the subgraph induced by the set of all vertices that remain to be dominated. The longest path P in C' that can be dominated by k' vertices has length $3k' - 1$. As by definition of C, each vertex $u \in V(C) \setminus V(C')$ is neighbored to a vertex $u' \in V(C')$, so for u there exists u' such that $\{u, u'\} \in E(C)$. Each of the endpoints of P may have a neighbor in $V(C) \setminus V(C')$ and therefore the longest path that may be dominated by k' vertices in C has length $3k' + 1$. □

If no termination condition holds, we proceed as follows. To hit all colors in $\mathcal{F}'$, we assign each $F \in \mathcal{F}'$ to either a component or backland in $\mathcal{C}' \coloneqq \mathcal{C} \cup \{B\}$. Such an assignment can be represented as a function f of type $\mathcal{F}' \to \mathcal{C}'$, which lets us refer to the set of F colors to be hit by $C \in \mathcal{C}$ as $f^{-1}(C)$. For the backland, accordingly, this set is given by $f^{-1}(B)$.

To check for each possible assignment of F colors to the elements of $\mathcal{C}'$ if a generalized dominating set exists in G', we iterate over all functions f in $\{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function }\}$.

For a given f, we repeat the following procedure for each $C \in \mathcal{C}$. We intersect each color with the set of vertices of C to get $H_C \coloneqq H' \cap V(C)$ and $\mathcal{F}_C \coloneqq \{F_i \cap V(C) \mid F_i \in f^{-1}(C)\}$. Any colored vertex $v \notin V(C)$ can be deleted for recursive calls of `RGDS`, because such a vertex cannot be used by C to hit a color in $\mathcal{F}_C$. Since $\mathcal{C}$ is the set of components of $G'[N[W]] = G'[N[V(G) \setminus H]]$, we ensure the requirement of `RGDS` that $H, F_1, \ldots, F_l \subseteq N[V(G) \setminus H]$ for each $C \in \mathcal{C}$.

Then, we compute the minimum number of vertices required for C to contain a generalized dominating set given H_C and $\mathcal{F}_C$ by recursively calling `RGDS` with increasing values for the parameter k. We denote this minimum number of vertices as $k_{f^{-1}(C),C}$ and write D_C to refer to the corresponding dominating set returned by `RGDS`. Another two termination conditions related to the components in $\mathcal{C}$ exist.

Claim 3.4. *Let $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function }\}$. According to the assignment of colors given by f, there is no generalized dominating set for $(G', H', \mathcal{F}', k')$ if either of the following holds.*
1) There exist $C \in \mathcal{C}$ and $F \in f^{-1}(C)$ such that $F \neq \emptyset$ but $F \cap V(C) = \emptyset$.
2) It exists a component $C \in \mathcal{C}$ such that $(C, H_C, \mathcal{F}_C, k')$ is not an instance of the dominating set problem.

Proof. A component $C \in \mathcal{C}$ cannot hit a color $F \in f^{-1}(C)$ if $F \not\subseteq V(C)$, which leads to the first termination condition. If $|D_C| > k'$, then $(G', H', \mathcal{F}', k')$ cannot be an instance of the generalized dominating set problem, since component C alone requires more than k' vertices to contain a generalized dominating set. That leads to the second termination condition. □

After having computed D_C for each $C \in \mathcal{C}$, we move on to the backland B. B is the set of vertices that are not in any component $C \in \mathcal{C}$, which means that all vertices in B are already dominated. As already indicated, however, B cannot be discarded, since it is required to hit the colors $\mathcal{F}_B \coloneqq \{F_i \cap B \mid F_i \in f^{-1}(B)\}$.

Claim 3.5. *A minimum set of vertices H_B hitting each color $F \in f^{-1}(B)$ is given by a minimum hitting set of $f^{-1}(B)$.*

Proof. This follows immediately from the definition of the minimum hitting set problem. □

In general the hitting set problem is NP-complete [29]. However, note that for the initial call of `RGDS` we have $\mathcal{F} = \emptyset$ and that a color F is added to $\mathcal{F}$ only in the branch $v \notin D'$ (which will be explained below). In the proof of efficiency of `RGDS` in Section 3.5 we construct a search tree whose degree and depth are bounded in terms of k. Hence, also the complexity of the hitting set instance that arises in `RGDS` can be bounded in terms of k.

As with the components in C, there exist termination conditions that allow us to immediately continue to the next function f.

Claim 3.6. *Let $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function }\}$. According to the assignment of colors given by f, there is no generalized dominating set for $(G', H', \mathcal{F}', k')$ if either of the following holds.*
1) There exists $F \in f^{-1}(B)$ such that $F \neq \emptyset$ but $F \cap B = \emptyset$.
2) There exists no hitting set H_B for $f^{-1}(B)$ with $|H_B| \leq k'$.

Proof. Backland cannot hit a color $F \in f^{-1}(B)$ with $F \not\subseteq B$, which leads to to the first termination condition. If backland requires more than k' vertices to hit all colors in $f^{-1}(B)$, there cannot exist a generalized dominating set for $(G', H', \mathcal{F}', k')$. That leads to the second termination condition. In either case, f assigns colors such that $(G', H', \mathcal{F}', k')$ cannot be an instance of the generalized dominating set problem. □

If no termination condition holds, we now have all intermediate results to verify whether, given f, all vertices in W can be dominated and all colors in $\mathcal{F}'$ hit. For this purpose, we sum up the number of vertices required to dominate the components in $\mathcal{C}$ and B and compare the result to k', the maximum number of vertices that may be added to D''. Let $k_{f^{-1}(B),B} \coloneqq |H_B|$, where H_B is a minimum hitting set of $f^{-1}(B)$. In case $(\sum_{C \in \mathcal{C}} k_{f^{-1}(C),C}) + k_{f^{-1}(B),B} > k'$, there is no generalized dominating for the assignment of colors in $\mathcal{F}'$ according to f. Hence, we move on to the next function $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function }\}$.

Otherwise, if the sum of vertices required per component and backland is not greater than k', we have found a generalized dominating set for $(G', H', \mathcal{F}', k')$. In this case, `RGDS` shall return a generalized dominating set, so the return value is $D \cup \{v\} \cup \bigcup_{C \in \mathcal{C}} D_C \cup H_B$. To $D \cup \{v\}$, the vertices that have already been added to the generalized dominating set, we add the generalized dominating sets of the components in $\mathcal{C}$, that is $\bigcup_{C \in \mathcal{C}} D_C$. Furthermore, we add H_B, the vertices used to hit all colors in $\mathcal{F}_B$.

Claim 3.7. *If no $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function}\}$ allows us to find a generalized dominating set D' for $(G', H', \mathcal{F}', k')$ with $v \in D'$, then there is no generalized dominating set with $v \in D'$.*

Proof. By iterating over all functions $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function}\}$, we test all possible assignments of colors in $\mathcal{F}'$ to the elements in $\mathcal{C}'$. □

In the next step we are going to search for a generalized dominating set assuming $v \notin D'$. Except for an additional termination condition and some adjustments to the colors passed to `Solve`, the procedure is the same as for the branch $v \in D'$. H' and $\mathcal{F}'$ store information about vertices that are already dominated and colors that need to be hit. Hence, if v is no longer added to D', we have to modify H' and $\mathcal{F}'$. Let us begin with the additional termination condition.

Claim 3.8. *If $N(v) = \emptyset$ and $v \notin H$ there is no generalized dominating set D' for $(G', H', \mathcal{F}', k')$ with $v \notin D'$.*

Proof. If $v \notin H$ and $v \notin D'$, then a vertex $u \in N(v)$ has to be added to D', otherwise v would not be dominated. However, if $N(v) = \emptyset$ this is not possible. □

With respect to the colors, we note that we can set $H' = H$. If v is not added to the generalized dominating set, none of its neighbors becomes dominated at this point. If $N(v) \neq \emptyset$ and $v \notin H$, we must introduce a new color $F_{l+1} \coloneqq N(v)$, to enforce that a vertex which can dominate v is added to D' in the future. In this case, we set $\mathcal{F}' = \mathcal{F} \cup \{N(v)\}$. Otherwise, if $v \in H$, we do not necessarily have to add a neighbor of v to D'. Accordingly, $\mathcal{F}$ may remain unchanged and in this case we have $\mathcal{F}' = \mathcal{F}$.

As we have not added a vertex to the generalized dominating set, we get $k' = k$. Then, repeating our procedure of the branch $v \in D'$, we call `Solve` $(\mathcal{C}, H', \mathcal{F}', k', D, V(G'))$ to find a generalized dominating set. If we have not found a generalized dominating set after trying $v \in D'$ and $v \notin D'$, $(G, H, \mathcal{F}, k)$ is not a solution of the generalized dominating set problem.

3.4 Correctness of algorithm `RGDS`

We prove the correctness of `RGDS` (Algorithm 1) by induction over the graph size $n \coloneqq |V(G)|$. Our approach is to show that `RGDS` returns a generalized dominating set if and only if such a set exists in $(G, H, \mathcal{F}, k)$. If $(G, H, \mathcal{F}, k)$ is not an instance of the generalized dominating set problem, `RGDS` returns `Null`. For the *base case* of the induction we have $n = |V(G)| = 1$ and for the initial call of `RGDS` we can set $D = \emptyset$.

Claim 3.9. *Let $G = (\{v\}, \emptyset)$ (that is $|V(G)| = 1$). Let $H, F_1, \ldots, F_l \subseteq N[V(G) \setminus H]$ be colors, $\mathcal{F} = \{F_1, \ldots, F_l\}$, $k \in \mathbb{N}$. Let $D = \emptyset$ for the initial call of* `RGDS`. `RGDS` *returns a generalized dominating set D' for $(G, H, \mathcal{F}, k)$ if and only if $(G, H, \mathcal{F}, k)$ is an instance of the generalized dominating set problem. Otherwise,* `RGDS` *returns* `Null`.

Proof. We make a distinction of the following two cases.

Case $k = 0$. $(G, H, \mathcal{F}, 0)$ is an instance of the generalized dominating set problem if and only if v is already dominated and no color $F \in \mathcal{F}$ remains to be hit. `RGDS`[2] tests these properties by verifying $(V(G) \setminus H) \cup \bigcup_{F \in \mathcal{F}} F = \emptyset$. If it holds, $D' = \emptyset$ is returned which, in this case, is a generalized dominating set for $(G, H, \mathcal{F}, 0)$. Otherwise, $(G, H, \mathcal{F}, 0)$ cannot contain a generalized dominating set. Due to $k = 0$ no vertex may be added to D', however v is not yet dominated if $v \notin H$ or a color remains to be hit if $\mathcal{F} \neq \emptyset$. Accordingly `RGDS` returns `Null`.

Case $k > 0$. If $k > 0$, a generalized dominating set for $(G, H, \mathcal{F}, k)$ is always given by $\{v\}$. In case $v \in H$ and $\mathcal{F} = \emptyset$, however, we can simply return $D' = \emptyset$ (see Case $k = 0$). Otherwise, `RGDS` returns $D' = D \cup \{v\} = \{v\}$, which is a generalized dominating set because $F_1, \ldots, F_l \subseteq V(G)$. □

For the *inductive step*, let G be a graph of size $n > 1$, $H, F_1, \ldots, F_l \subseteq V(G)$, $\mathcal{F} = \{F_1, \ldots, F_l\}$ and $k \in \mathbb{N}$. For the initial call of `RGDS`, again, we can set $D = \emptyset$.

Claim 3.10. *Let G be a graph with $n = |V(G)| > 1$. Let $H, F_1, \ldots, F_l \subseteq N[V(G) \setminus H]$ be colors, $\mathcal{F} = \{F_1, \ldots, F_l\}$, $k \in \mathbb{N}$. Let $D = \emptyset$ for the initial call of* `RGDS`. `RGDS` *returns a generalized dominating set D' for $(G, H, \mathcal{F}, k)$ if and only if $(G, H, \mathcal{F}, k)$ is an instance of the generalized dominating set problem. Otherwise,* `RGDS` *returns* `Null`.

Proof. If $k = 0$, we can proceed as in the case $k = 0$ of the *base case*. That is, if all vertices are already dominated and $\mathcal{F} = \emptyset$, then $D = \emptyset$ is a generalized dominating set for $(G, H, \mathcal{F}, k)$ and `RGDS`[2] returns D. Otherwise `RGDS` returns `Null`.

If $k > 0$, we choose[2] a vertex $v \in V(G)$. First, we analyze the branch $v \in D'$ as described in Section 3.3. We set $H' = H \cup N(v)$, $\mathcal{F}' = \mathcal{F} \setminus \{F \in \mathcal{F} \mid v \in F\}$ and $B = V(G) \setminus \bigcup_{C \in \mathcal{C}} V(C)$. Let $G' = G - v$, we then compute $W = V(G') \setminus H$ and $\mathcal{C} = \{C \mid C \text{ component of } G'[N[W]]\}$ in `RGDS`$[5-7]$. According to Claim 3.2, the components in $\mathcal{C}$ are the subgraphs to be passed to recursive calls of `RGDS`. We call `Solve`$(\mathcal{C}, H', \mathcal{F}', k', D \cup \{v\}, V(G'))$ to find a generalized dominating set D'' for $(G', H', \mathcal{F}', k')$.

In `Solve`$[2-4]$ we return `Null` if any of the termination conditions described in Claim 3.3 holds. If no termination condition holds, we try each possible assignment of colors to elements in $C' = \mathcal{C} \cup \{B\}$ by iterating over all functions $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function}\}$.

Given such a function f, we proceed as follows. The first termination condition described in Claim 3.4 is tested in `Solve`[8]. If it holds, we can immediately discard f and continue with the next function. We repeat the same for the first termination condition of Claim 3.6 in `Solve`[14]. For each component $C \in \mathcal{C}$ and the colors it is required to hit, that is $f^{-1}(C)$, we determine a minimum generalized dominating set as described in Section 3.3. Since we have removed v from G and each $C \in \mathcal{C}$ is a subgraph of $G - v$, it holds that $|V(C)| < |V(G)|$ for all components in $\mathcal{C}$. Hence, as per *induction hypothesis*, `RGDS` returns a generalized dominating set D_C for $(C, H_C, \mathcal{F}_C, k_{f^{-1}(C),C})$ if and only if $(C, H_C, \mathcal{F}_C, k_{f^{-1}(C),C})$ is an instance of the generalized dominating set problem. Otherwise, `RGDS` returns `Null` and we can continue to the next function f according to the second termination condition in Claim 3.4.

With respect to backland B, we determine a minimum hitting set H_B for $f^{-1}(B)$ (Claim 3.5). If either of the termination conditions described in Claim 3.6 holds, we discard f and continue to the next function.

If none of the termination condition held, all intermediate results are now available to test if $(G', H', \mathcal{F}', k')$ is an instance of the generalized dominating set problem. From the *induction hypothesis* and Claim 3.5 it follows that all vertices in W are dominated and all colors in $\mathcal{F}'$ hit. The number of vertices required for this is $|D''| = (\sum_{C \in \mathcal{C}} k_{f^{-1}(C),C}) + k_{f^{-1}(B),B}$. If $|D''| < k'$ we have found a generalized dominating set for $(G', H', \mathcal{F}', k')$, that is $D'' = \bigcup_{C \in \mathcal{C}} D_C \cup H_B$. As we are in the branch $v \in D'$ the generalized dominating set for $(G, H, \mathcal{F}, k)$ is given by $D' = D \cup \{v\} \cup D''$.

Otherwise, if $|D''| > k$, no generalized dominating set exists for $(G', H', \mathcal{F}', k')$ given f, as we have determined minimum generalized dominating sets for all components in C and a minimum hitting set for backland B. Hence, we move on to the next function $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function }\}$.

After testing all functions of type $\mathcal{F}' \to \mathcal{C}$ without finding a generalized dominating set D'' with $|D''| \leq k'$, we can conclude that no such generalized dominating set exists for $v \in D'$ (Claim 3.7) and move on to the branch $v \notin D'$.

[2] *How* v is chosen is crucial for the efficiency of `RGDS` and will be explained in Section 3.5. For now, assume v is chosen randomly. This allows us to focus on the correctness of the algorithm.

The additional termination condition for the branch $v \notin D'$ introduced in Claim 3.8 is tested in `RGDS`[14]. The colors are adjusted as described in Section 3.3, that is $H' = H$. After modifying H we recalculate W, $\mathcal{C}$ and k'. The value of $\mathcal{F}'$ depends on the following condition. If $v \notin H$ and $N(v) \neq \emptyset$ we set $\mathcal{F}' = \mathcal{F} \cup \{N(v)\}$ to ensure that v will be dominated in the future. Otherwise $\mathcal{F}' = \mathcal{F}$. We then repeat the procedure described above by calling `Solve`$(\mathcal{C}, H', \mathcal{F}', k', D, V(G'))$.

If `Solve` returns $D'' \neq$ `Null`, a generalized dominating set for $(G, H, \mathcal{F}, k)$ is given by $D' = D \cup D''$, as we are in the branch $v \notin D'$. However, if `Solve` returns `Null`, there exists no generalized dominating set with $v \notin D'$.

For both $v \in D'$ and $v \notin D'$ we have tried all possible assignments of colors in $\mathcal{F}'$ to components in $\mathcal{C}'$. If a generalized dominating set D' exists for $(G, H, \mathcal{F}, k)$, it is found and `RGDS` returns D'. If neither for $v \in D'$ nor $v \notin D'$ a generalized dominating set exists, `RGDS` returns `Null`. It follows that `RGDS` returns a generalized dominating set D' for $(G, H, \mathcal{F}, k)$ if and only if $(G, H, \mathcal{F}, k)$ is an instance of the generalized dominating set problem. Otherwise `RGDS` returns `Null`. □

3.5 Efficiency of algorithm `RGDS`

We show that the algorithm `RGDS` is efficient by constructing a search tree of bounded degree and bounded depth. The boundary of the degree is a function of k, which is the size of the solution. The depth of the search tree is bounded by the number of rounds Splitter needs to win the splitter game on input graph G. As we will see, also the number of rounds is a function of k. It follows that the algorithm `RGDS` is efficient.

Bounded degree. The input parameters of `RGDS` are $(G, H, \mathcal{F}, k, D)$ and we write D' to refer to the algorithm's output. Given graph G and our choice for vertex $v \in V(G)$ that will be removed from G, we have one branch for $v \in D'$ and one for $v \notin D'$. Except for different colorings that may result in different sets of components of $G[N[W]]$, the procedure in each of the two branches is the same. In either branch we call `Solve`$(\mathcal{C}, H', \mathcal{F}', k', D'', V(G'))$, whereas $H', \mathcal{F}', k', D''$ are the input parameters modified according to $v \in D'$ respectively $v \notin D'$.

`Solve` immediately returns `Null` if $|\mathcal{C}| > k'$, so in the following we assure $|\mathcal{C}| \leq k'$. If $v \in D$, then k is decremented by one, otherwise it is left unchanged, therefore $k - 1 \leq k' \leq k$. We then branch once for each possible assignment of colors in $\mathcal{F}'$ to components in $\mathcal{C}' \coloneqq \mathcal{C} \cup \{B\}$, that is once for each function $f \in \{f : \mathcal{F}' \to \mathcal{C}' \mid f \text{ function}\}$.

With respect to the cardinality of $\mathcal{F}'$ the following holds. As we call `RGDS` to decide if (G, k) is an instance of the dominating set problem, according to Lemma 3.3.1 we may assume that $\mathcal{F} = \emptyset$ for the initial call of `RGDS`. In each recursive call at most one color is added to $\mathcal{F}$ if $v \notin D'$. Assume for now (it will be shown below) that the depth of the search tree is bounded by a function of k. Let h be such a function, then $|\mathcal{F}'| \leq h(k)$. Hence, the number of functions of type $\mathcal{F}' \to \mathcal{C}'$ is bounded by $|\mathcal{C}'|^{|\mathcal{F}'|} = (k+1)^{h(k)}$.

Given a function f, `Solve` calculates a minimun generalized dominating set for each component in $\mathcal{C}$. If there exists a component $C \in \mathcal{C}$ for which no generalised dominating

set D_C with $|D_C| \leq k'$ exists, then f is rejected. Therefore, we can conclude that the degree in the search tree is bounded by $(k+1)^{h(k)}$.

Bounded depth. Let $(G, H, \mathcal{F}, k, D)$ be the parameters passed to `RGDS` and D' the output of `RGDS`. The algorithm terminates if $k = 0$ or if $|V(G)| \leq 1$. Splitter wins the ℓ-round radius-r splitter game if there exists $i \in \mathbb{N}$ such that $G_{i+1} = \emptyset$. According to Theorem 2.5.1, for every positive integer r there is an integer ℓ such that Splitter wins the ℓ-rounds radius-r splitter game on nowhere dense graphs. It follows that ℓ is an upper bound for the depth of the search tree if we chose the vertex v to be deleted in `RGDS` according to a winning strategy for splitter.

As mentioned earlier (Section 2.5.3), ℓ can be expressed as a function of r, however that function is huge for a general nowhere dense class of graphs. According to Theorem 2.5.3, splitter wins the ℓ-round radius-r splitter game if $\ell = \mathrm{wcol}_{2r}(G)$. In [41] it is shown that $\mathrm{wcol}_r(G) \leq \binom{t}{2}^r \cdot (2r+1)$ if G excludes K_t as a minor. This provides an upper bound for the depth of the search tree that may result in *acceptable* running times of our algorithm in practice for graphs excluding K_t as a minor.

Let us now relate our algorithm to the splitter game and the splitter game radius r to the parameter k. We have shown in Section 3.4 that the longest path that can be dominated by k vertices in a component of $G[N[W]]$ has length $3k+1$. Hence, if we play the radius $r' = 3k+1$ splitter game, we ensure that Splitter can always follow the winning strategy described in Theorem 2.5.3. We follow this winning strategy in `RGDS` by choosing v (the vertex to be deleted) as the minimum element of an order $\leq$ that witnesses $\mathrm{wcol}_{2r'}(G) \leq \ell$. It follows that, in terms of k, the depth of the search tree is bounded by $\mathrm{wcol}_{6k+2}(G) \leq g(6k+2)$ for a class of graphs with bounded expansion and a function g (Theorem 2.5.2).

4 Empirical analysis of RGDS

4.1 Implementation

We have implemented the algorithm RGDS in *C++* to analyze its practical running time. To store graphs and for some graph routines we used the Boost Graph Library (BGL) [1]. We rely on the library of Court [14] to compute weak coloring numbers and the orders of vertices according to which we choose v, that is the vertex to be removed in RGDS.

Kernelization. With regard to the implementation of the data reduction rules presented in Section 2.4, we follow the same approach Alber, Fellows, and Niedermeier [10] have taken in their empirical analysis. Let G be a graph and v be a vertex for which the reduction rules introduce a gadget vertex v' to enforce that v will be added to a dominating set. Instead of adding v' and the edge $\{v, v'\}$ to the graph, we remove v and all edges incident to v. Moreover, we add v to D_{PP}, the set of vertices that must be added to a dominating set according to the preprocessing rules. By adding v to the dominating set, the neighborhood of v becomes dominated and therefore we add $N(v)$ to H_{PP}. We write H_{PP} to refer to the set of vertices that are dominated by D_{PP}, that is for each $h \in H_{PP}$ there exists $d \in D_{PP}$ such that $\{h, d\} \in E(G)$.

Let G' be the graph resulting from the application of the reduction rules, D_{PP} the sets of vertices added to a dominating set according to the reduction rules and H_{PP} be the set of vertices dominated by D_{PP}. If (G, k) is the original dominating set problem, we call RGDS$(G', H_{PP}, \emptyset, k - |D_{PP}|)$ according to Lemma 3.3.1.

Rule 2 Case 1.1, however, requires special treatment. Let $v, w \in V(G)$ $(v \neq w)$ be two vertices subject to Rule 2 Case 1.1. The reduction rule states that we must add *at least one* of the vertices to a dominating set. That means we do not know if it is only one or both vertices which must be added to D_{PP}. In case it is only one, we do not know if it is v or w.

To overcome this issue, we leave G, D_{PP} and H_{PP} unchanged when dealing with Rule 2 Case 1.1 and add the gadget vertices $\{z, z'\}$ and their incident edges according to the rule. Let D' be the generalized dominating set returned by RGDS$(G', H_{PP}, \emptyset, k - |D_{PP}|)$. After the application of the second data reduction rule, D' may contain gadget vertices. To fix this, we repair D' by removing $\{z, z'\}$ from D' and adding the smallest subset of $\{v, w\}$ to D' such that D' becomes a dominating set of (G, k).

Multithreading. Our implementation of RGDS supports multithreading. The number of cores available for RGDS is passed on as a command line argument when executing the algorithm. Let n_C be the number of cores passed on via the command line. The

maximum number of threads that can be running concurrently at any time is then given by n_C.

Multithreading comes into play when calculating minimum generalized dominating sets for the components in $\mathcal{C}$. The number of currently active threads n_{act} is compared to n_C and $n_C - n_{act}$ new threads will be started. Each thread receives an equal share of the components in $\mathcal{C}$ for which it calculates minimum generalized dominating sets. If $n_C = 1$ or $n_C - n_{act} = 0$, no new threads will be started and all components will be handled in the current thread.

4.2 Results

4.2.1 KONECT

To analyze the running time of our RGDS implementation we calculated dominating sets of graphs provided by KONECT (the Koblenz Network Collection) [3]. KONECT is a collection of graphs from various sources. Graphs collected in KONECT stem from social media, transportation, hyperlink networks and social sciences, for example. Currently KONECT provides 241 graphs of which 57 are undirected. We have started RGDS on a subset of the undirected graphs.

We focus on simple graphs (i.e. no loops and no multiple edges between the same pair of vertices) and try to cover graphs arising from a large variety of applications. However, we included all road networks as the corresponding graphs can be expected to be mainly planar due to the properties of roads. In particular, we ran RGDS on the following KONECT graphs: Amazon MDS, Brightkite, CAIDA, Facebook friendships, Facebook NIPS, Flickr, Hamsterster full, Jazz musicians, PDZBase, Pretty Good Privacy, Roadnet California, Roadnet Pennsylvania, Roadnet Texas, Route views, US power grid, WordNet.

For the execution of RGDS we use the supercomputer of the Logics and Semantics research group from the Technical University Berlin. It consists of 28 CPUs which we used exclusively during the execution of our tests, that is no other tasks were running. We decided to run RGDS on as many graphs as possible and, hence, did not utilize multithreading because there are more undirected graphs in KONECT than we have CPUs available on our computer.

The size of the KONECT graphs ranges from 16 vertices and 56 edges for animal genealogies to approximately 5M vertices and 50M edges for Livejournal links. Below we give an overview of graphs on which the algorithm terminated within three weeks and provide statistics related to the running time. All graphs and more statistics about them are freely available on the KONECT website [3].

For the application of our algorithm on KONECT graphs we did not apply data reduction Rule 2 (except for the Facebook ego-network) due to the following reason. In our current implementation of Rule 2 it is required to store the neighborhood sets $N_1(v, w)$, $N_2(v, w)$ and $N_3(v, w)$ for all pairs of vertices $v, w \in V(G)$ of a graph G, that is for $\binom{|V(G)|}{2}$ pairs of vertices. All neighborhood sets need to be calculated and stored since we iteratively delete vertices as described in Section 4.1. As the number of vertices in a

graph exceeds several thousand, it is no longer possible to hold all neighborhood sets in memory[1].

We are now going to introduce the KONECT graphs on which our algorithm terminated within three weeks. Each graph was handled in a single thread on an Intel Xeon E5-2697 (2.60 GHz) core on which no other tasks ran during our tests. We briefly describe each graph and explain how vertices and edges are interpreted in the application from which the graphs arise. Moreover, we provide data in two tables for each graph.

The first table contains the number of vertices and edges and the largest value for $t \in \mathbb{N}$ encountered in the approximation algorithm of weak coloring numbers such that K_t is a minor of the graph. Note that this is not necessarily the maximum t such that K_t is a minor. The second table contains statistics related to the execution of RGDS. By k we refer to the size of the minimum dominating set. We give running times of data reduction rules (column D.R. time) and of RGDS (RGDS time). With respect to data reduction, we list the number of vertices remaining in the graph after data reduction (V. rem.) and the number of vertices that were deleted (V. del.). Moreover, we give the weak coloring number of the graph on which RGDS was applied (wcol_{6k+2}), that is the weak coloring number of the reduce graph if data reduction Rule 1 or Rule 2 was applied

Facebook ego-network[2][6]. This graph is the ego-network of a Facebook user. An ego-network represents the connections between the friends of a given Facebook user [34]. We refer to this user as the ego. The vertices in an ego-network are the friends of the ego. For each friendship between two friends of the ego, there exists an edge in the ego network [6].

Table 4.1: FB ego-network: descriptive statistics

Vertices	Edges	largest t in wcol algo s.t. K_t minor
2'888	2'981	3

Table 4.2: FB ego-network: results

$k :=$ size of minimum dom. set $= 10$					
Data Reduction	RGDS time	D. R. time	V. rem.	V. del.	wcol_{6k+2}
None	163 sec	0	2'888	0	13
Rule 1	< 1 sec	1 sec	11	2'877	1
Rule 2	5 sec	≈ 1 h	10	2'878	1

PGP Web of Trust [7]. PGP is the well known cryptography program Pretty Good

[1] It would be possible to circumvent this limitation by a more advanced implementation of Rule 2 (see Section 5.2). This would be beyond the scope of this thesis, however, as our focus is not on data reduction.

[2] Note that this graph's description provided by KONECT is not consistent. At one point it is said to be directed, while at another point it is said to be undirected. According to the original source, this graph is undirected [8].

Privacy. The PGP Web of Trust is the network of PGP users who have signed the public keys of other users they trust to truly represent who they claim to be. This network includes only edges between users that have signed their keys mutually in the PGP Web of Trust as of July 2001 [12]. Only the largest connected component is included in this graph, the remainder of the graph was discarded and is not available on KONECT.

Table 4.3: PGP WoT: descriptive statistics

Vertices	Edges	largest t in wcol algo s.t. K_t minor
10'680	24'316	17

Table 4.4: PGP WoT: results

$k :=$ size of minimum dom. set = 2'712					
Data Reduction	RGDS time	D. R. time	V. rem.	V. del.	wcol_{6k+2}
None	not terminated	0	10'680	0	719
Rule 1	< 1 sec	< 1 sec	1'963	8'717	93
Rule 2	not applied				

CAIDA autonomous systems [5]. The Center for Applied Internet Data Analysis (CAIDA) is an organization initiated by various entities from commercial, government and research sectors to promote collaboration in the engineering and maintenance of the internet's infrastructure [2]. Caida collected this network of autonomous systems in the internet, i.e. internet service providers (ISPs), and their interactions. A vertex represents an autonomous system and two vertices are connected by an edge if internet traffic was routed between the two corresponding autonomous systems [5]. The data was collected in 2007.

Table 4.5: CAIDA: descriptive statistics

Vertices	Edges	largest t in wcol algo s.t. K_t minor
26'475	53'381	24

Table 4.6: CAIDA: results

$k :=$ size of minimum dom. set = 2400					
Data Reduction	RGDS time	D. R. time	V. rem.	V. del.	wcol_{6k+2}
None	not terminated	0	26'475	0	833
Rule 1	7 sec	21 sec	9'398	17'077	18
Rule 2	not applied				

Brightkite [4]. Brightkite was a location based social network founded in 2007 that

went offline in 2011. It allowed users to befriend other users, share locations and view posts published in relation to locations. Vertices in this graph represent users and edges exist between two users that have mutually added each other as friends.

Table 4.7: Brightkite: descriptive statistics

Vertices	Edges	largest t in wcol algo s.t. K_t minor
58'228	214'078	80

Table 4.8: Brightkite: results

$k :=$ size of minimum dom. set = 13'531					
Data Reduction	RGDS time	D. R. time	V. rem.	V. del.	wcol_{6k+2}
None	not terminated	0	58'228	0	10'374
Rule 1	$\approx$ 35 h	6 sec	19'450	41'151	3'340
Rule 2	not applied				

For the remaining graphs that we tested, the computations have not yet terminated after running for more than three weeks. Given the size of these graphs and because the dominating set problem is W[2] complete, this result is not surprising. Our tests on random graphs excluding a K_5 as a minor with high probability provide an explanation of these running times.

4.2.2 Random graphs excluding K_5 as a minor

We have analyzed the running time of our algorithm on random graphs that exclude K_5 as a minor with high probability. As indicated in Section 2.5.2, graphs whose edge density is bounded by $(\alpha + o(1)) \cdot t\sqrt{\ln t}$ where $\alpha = 0.39$ exclude K_t as a minor with high probability [40]. We construct random graphs by iterating over the adjacency matrix and inserting edges with a probability such that the corresponding extremal edge density is not exceeded. The algorithm for weak coloring numbers produced good orders, as expected. The results and some descriptive statistics of the random graphs are provided below.

For $n = 10, 20$ we have run our algorithm on 100 graphs. Tables 4.9 and 4.10 provide data averaged over 100 random graphs each for graphs which have not been preprocessed and for graphs to which data reduction Rule 1 respectively Rule 2 has been applied (see column D.R.). We list the number of edges ($|E|$) and the percentage of vertices that was removed from the graphs due to data reduction ($|V|$ del.). Running times are given for data reduction (D.R. time), the RGDS algorithm (RGDS time) and the wcol algorithm (wcol time). Column |DomSet| contains the average size of the dominating sets and column wcol_{6k+2} the average of the corresponding weak coloring numbers, where for each graph k refers to the size of its minimum dominating set.

Table 4.9: Averages of 100 random graphs expected to excl. K_5 with $|V(G)| = 10$

D.R.	$\|E\|$	D.R. time	$\|V\|$ Del.	RGDS time	|DomSet|	wcol time	wcol_{6k+2}
None	29	0	0%	< 1 sec	1.84	< 1 sec	8.79
Rule 1	28	< 1 sec	16.3%	< 1 sec	1.88	< 1 sec	8.71
Rule 2	28	< 1 sec	9.71%	< 1 sec	1.87	< 1 sec	8.81

Table 4.10: Averages of 100 random graphs expected to excl. K_5 with $|V(G)| = 20$

D.R.	$\|E\|$	D.R. time	$\|V\|$ Del.	RGDS time	|DomSet|	wcol time	wcol_{6k+2}
None	60	0	0%	15.09	3.68	< 1 sec	15.65
Rule 1	60	< 1 sec	1.5%	22.82	3.81	< 1 sec	15.53
Rule 2	60	< 1 sec	1.8%	6.67	3.64	< 1 sec	15.64

In Table 4.11 we provide the same data as above, however not averaged over 100 graphs but for individual graphs. Each row refers to a single graph and column D.R. indicates whether data reduction was applied. Note that RGDS running times increase dramatically for $|V(G)| = 30$. On other random graphs that exclude K_5 as a minor with high probability and with $|V(G)| = 30$ the algorithm did not terminate within a week.

Table 4.11: Random graphs expected to exclude K_5 with $|V(G)| = 30$

D.R.	$\|E\|$	D.R. time	$\|V\|$ Del.	RGDS time	|DomSet|	*wcol* time	$wcol_{6k+2}$
None		0	0%	11.28 h	5	< 1 sec	22
None		0	0%	10.27 h	5	< 1 sec	22
Rule 1		< 1 sec	0%	33.26 h	6	< 1 sec	24
Rule 2		< 1 sec	0%	38.41 h	5	< 1 sec	23
Rule 2		< 1 sec	0%	114.54 h	6	< 1 sec	22

5 Conclusion and future work

5.1 Conclusion

We introduced the efficient fixed-parameter algorithm `RGDS` to decide the dominating set problem (G, k) for a graph G that is nowhere dense and $k \in \mathbb{N}$. The efficiency of `RGDS` was proven by constructing a search tree whose degree and depth are bounded by $f(k)$, where f is a computable function. Our algorithm is recursive and terminates when $k = 0$ or $|V(G)| \leq 1$. In each call of `RGDS` a vertex is deleted until the recursive base case is reached. The base case $|V(G)| = 0$ corresponds to the position where Splitter wins the ℓ-round radius-r splitter game on G. Hence, the depth of the search tree is bounded by the number of rounds required such that Splitter can win the ℓ-round radius-r splitter game.

According to Theorem 2.5.3, Splitter wins if $\text{wcol}_{2r} \leq \ell$, which gives us a boundary for the depth of the search tree in terms of weak coloring numbers. In this case, a winning strategy for Splitter is to pick in each round the minimum element of an order $\leq$ on $V(G)$ that witnesses $\text{wcol}_{2r}(G) \leq \ell$. We follow this approach in `RGDS` when picking v, the vertex to be removed from input Graph G. Hence, for every class of graphs on which we know how to efficiently compute high quality orderings, we can hope that our algorithm terminates quickly. This is the case for classes of graphs with bounded expansion and classes of graphs that exclude a fixed minor K_t.

We have implemented the algorithm `RGDS` in *C++* and let it run on graphs provided by KONECT and on random graphs that exclude K_5 as a minor with high probability. In the empirical analysis, each graph was handled in a single thread on an Intel Xeon E5-2697 (2.60 GHz) core on which no other tasks ran during our tests. Within three weeks, our algorithm terminated on four KONECT graphs, on three of them even in remarkably short time. For example, `RGDS` computes a minimum dominating set of size 10 for a Facebook ego-network graph with 2'888 vertices and 2'981 edges in less than three minutes. In this respect, our results are encouraging. For the remaining KONECT graphs on which we have started our algorithm, however, it has not terminated within three weeks.

Our tests on random graphs excluding K_5 as minor with high probability provide an explanation for the extended running times. We constructed 100 random graphs for graph sizes $n := V(G) = 10, 20$ and computed for each graph a minimum dominating set as well as wcol_{6k+2}, where k is the size of a minimum dominating set. For $n = 10$, the running time for all 100 graphs combined is less than a second and the average wcol_{6k+2} is 8.9. The running time of the same operations for 100 graphs with $n = 20$ is 25 minutes (without data reduction) and the average wcol_{6k+2} is 15.65. Note the surprisingly large values of wcol_{6k+2}.

When moving on to $n = 30$, running times increase dramatically. Within a week, RGDS has terminated on five such random graphs that exclude K_5 as a minor with a high probability. Running times range from ten hours to almost five days. For each of these graphs, the size of a minimum dominating set is either five or six and the corresponding weak coloring numbers wcol_{6k+2} range from 22 to 24. These surprisingly high values of wcol_{6k+2} provide an explanation for the extended running times.

5.2 Future work

In this section we point out some of the possible optimizations of our implementation of RGDS. The implementation of these optimizations would be beyond the scope of this thesis.

Data reduction Rule 2. As mentioned in Section 4.1, our implementation of data reduction Rule 2 can only be applied to graphs with up to several thousand vertices. The exact number of vertices depends on the size of the neighborhood sets. If applied to graphs with more vertices than that, the process executing RGDS gets killed by the operating system because it exceeds its allowed memory usage. One way to overcome this limitation is storing neighborhood sets on disk instead of holding them in memory.

In our implementation, we immediately add vertices to which a reduction rule can be applied to the dominating set, instead of adding gadget vertices (see Section 2.4). For this purpose, all neighborhood sets need to be computed and stored before any vertex is deleted, which is the reason for the very high memory usage. Hence, the limitation related the graph size and Rule 2 could also be overcome by an implementation which does not require to hold all neighborhood sets in memory. This can be achieved, for example, by adding gadget vertices instead of removing vertices from the graph.

Storing return values of RGDS. Given the large number of recursive calls when searching for a minimum generalized dominating set of components in $\mathcal{C}$ in Solve[11], it might be an optimization to store return values of RGDS for parameters $(G, H, \mathcal{F}, k, D)$. It has to be tested, though, if finding a return value among all stored input tuples $(G, H, \mathcal{F}, k, D)$ will be faster than the computation of $\texttt{RGDS}(G, H, \mathcal{F}, k, D)$. For a speed up most likely input tuples have to be stored systematically such that the complexity of looking up a tuple of input parameters is less than $O(n)$, where n refers to the number of stored tuples $(G, H, \mathcal{F}, k, D)$. Moreover, it has to be tested if disk space will be sufficient to store all tuples for large graphs.

Eliminating functions from $\{f : \mathcal{F}' \to \mathcal{C}' \mid f$ **function**$\}$. In Solve[6] we iterate over all functions of type $\mathcal{F}' \to \mathcal{C}'$ to find an assignment of colors in $\mathcal{F}'$ to the elements in $\mathcal{C}'$ such that there exists a generalized dominating set for $(G', H', \mathcal{F}', k')$. Let f be such a function and let $f^{-1}(C)$ be the colors to be hit by a component $C \in \mathcal{C}'$ as given by f. If there is no generalized dominating set for C given f according to termination condition 2 in Claim 3.4, we can conclude that there is no generalized dominating set in

$(G', H', \mathcal{F}', k')$ for any function f' of type $\mathcal{F}' \to \mathcal{C}'$ with $f'^{-1}(C) = f^{-1}(C)$. Hence, we can optimize `RGDS` if we immediately discard any function f' with this property.

Moreover, of course, it would be nice to have more real world graphs similar to those provided by KONECT available to analyze the running time of the algorithm. In addition, if we had more cores available, we could obtain results for more graphs in a shorter period of time.

Bibliography

[1] *The Boost Graph Library: User Guide and Reference Manual.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[2] Caida. center for applied internet data analysis. `http://www.caida.org/home/`, 2016. Accessed: 2016-03-19.

[3] Konect. the koblenz network collection. `http://konect.uni-koblenz.de/`, 2016. Accessed: 2016-03-03.

[4] Konect. brightkite. `http://konect.uni-koblenz.de/networks/loc-brightkite_edges`, 2016. Accessed: 2016-03-03.

[5] Konect. caida autonomous systems. `http://konect.uni-koblenz.de/networks/as-caida20071105`, 2016. Accessed: 2016-03-03.

[6] Konect. facebook ego-network. `http://konect.uni-koblenz.de/networks/ego-facebook`, 2016. Accessed: 2016-03-03.

[7] Konect. pgp web of trust. `http://konect.uni-koblenz.de/networks/arenas-pgp`, 2016. Accessed: 2016-03-03.

[8] Snap. social circles: Facebook. `http://snap.stanford.edu/data/egonets-Facebook.html`, 2016. Accessed: 2016-03-03.

[9] 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(4):461–493, 2002.

[10] J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating set. *Journal of the ACM (JACM)*, 51(3):363–384, 2004.

[11] A. Atserias, A. Dawar, and P. G. Kolaitis. On preservation under homomorphisms and unions of conjunctive queries. *Journal of the ACM (JACM)*, 53(2):208–237, 2006.

[12] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70:056122, Nov 2004.

[13] B. Courcelle. Graph rewriting: An algebraic and logic approach. *Handbook of Theoretical Computer Science, volume B*, pages 193–242, 1990.

[14] A. Court. Empirical evaluation of approximation algorithms for graph colouring numbers. Master's thesis, Technische Universitt Berlin, 2016.

[15] A. Dawar and S. Kreutzer. Domination problems in nowhere-dense classes. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, pages 157–168, 2009.

[16] A. Dawar, M. Grohe, S. Kreutzer, and N. Schweikardt. Model theory makes formulas large. In *Automata, Languages and Programming*, pages 913–924. Springer, 2007.

[17] E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292–302, 2008.

[18] R. Diestel. *Graph Theory*. Springer, 3rd edition, 2006.

[19] R. G. Downey and M. R. Fellows. *Fixed-parameter tractability and completeness*. Cornell University, Mathematical Sciences Institute, 1992.

[20] P. G. Drange, M. S. Dregi, F. V. Fomin, S. Kreutzer, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, F. Reidl, F. S. Villaamil, S. Saurabh, S. Siebertz, and S. Sikdar. Kernelization and sparseness: the case of dominating set. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 31:1–31:14, 2016.

[21] Z. Dvořák, D. Král, and R. Thomas. Deciding first-order properties for sparse graphs. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 133–142. IEEE, 2010.

[22] J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM Journal on Computing*, 31(1):113–145, 2001.

[23] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM (JACM)*, 48(6):1184–1206, 2001.

[24] H. Gaifman. On local and non-local properties. In *Proceedings of the herbrand symposium, logic colloquium*, volume 81, pages 105–135, 1982.

[25] M. R. Garey and D. S. Johnson. A guide to the theory of np-completeness. *WH Freemann, New York*, 1979.

[26] M. Grohe, S. Kreutzer, and S. Siebertz. Deciding first-order properties of nowhere dense graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 89–98. ACM, 2014.

[27] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

[28] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

[29] R. M. Karp. *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations.*, chapter Reducibility among Combinatorial Problems, pages 85–103. Springer US, Boston, MA, 1972.

[30] H. A. Kierstead and D. Yang. Orderings on graphs and game coloring number. *Order*, 20(3):255–264, 2003.

[31] S. Kreutzer, M. Pilipczuk, R. Rabinovich, and S. Siebertz. Personal communication. 2016.

[32] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar. Evaluation of an mso-solver. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 55–63. Society for Industrial and Applied Mathematics, 2012.

[33] L. Libkin. *Elements of finite model theory*. Springer Science & Business Media, 2013.

[34] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *NIPS*, volume 2012, pages 548–56, 2012.

[35] J. Nešetřil and P. Ossona De Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

[36] J. Nešetřil and P. Ossona De Mendez. *Sparsity: Graphs, Structures, and Algorithms*, volume 28. Springer Science & Business Media, 2012.

[37] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[38] G. Philip, V. Raman, and S. Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Transactions on Algorithms (TALG)*, 9 (1):11, 2012.

[39] F. S. Roberts. *Graph theory and its applications to problems of society*. SIAM, 1978.

[40] A. Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001.

[41] J. van den Heuvel, P. Ossona de Mendez, R. Rabinovich, and S. Siebertz. On the generalised colouring numbers of graphs that exclude a fixed minor. *Electronic Notes in Discrete Mathematics*, 49:523–530, 2015.

[42] X. Zhu. Colouring graphs with bounded generalized colouring number. *Discrete Mathematics*, 309(18):5562–5568, 2009.