Kernelization in Almost Linear Time for Clustering into Bounded Vertex Cover Components

Sriram Bhyravarapu
Indian Institute of Technology Guwahati, India

Pritesh Kumar
The Institute of Mathematical Sciences, Chennai, India

Madhumita Kundu
University of Bergen, Norway

Shivesh K. Roy
The Institute of Mathematical Sciences, Chennai, India

Sahiba
Indian Institute of Technology Delhi, New Delhi, India

Saket Saurabh
The Institute of Mathematical Sciences, Chennai, India

Abstract

Motivated by the growing interest in graph clustering and the framework proposed during the Dagstuhl Seminar 23331, we consider a natural specialization of this general approach (as also suggested during the seminar). The seminar introduced a broad perspective on clustering, where the goal is to partition a graph into connected components (or "clusters") that satisfy simple structural integrity constraints – not necessarily limited to cliques.

In our work, we focus on the case where each cluster is required to have bounded vertex cover number. Specifically, a connected component C satisfies this condition if there exists a set $S \subseteq V(C)$ with $|S| \leq d$ such that $C - S$ is an independent set. We study this within the framework of the Vertex Deletion to d-Vertex Cover Components (Vertex Deletion to d-VCC) problem: given a graph G and an integer k, the task is to determine whether there exists a vertex set $S \subseteq V(G)$ of size at most k such that every connected component of $G - S$ has vertex cover number at most d. We also examine the edge-deletion variant, Edge Deletion to d-Vertex Cover Components (Edge Deletion to d-VCC), where the goal is to delete at most k edges so that each connected component of the resulting graph has vertex cover number at most d. We obtain following results.

1. Vertex Deletion to d-VCC admits a kernel with $\mathcal{O}(d^6k^3)$ vertices and $\mathcal{O}(d^9k^4)$ edges.
2. Edge Deletion to d-VCC, admits a kernel with $\mathcal{O}(d^4k)$ vertices and $\mathcal{O}(d^5k)$ edges.

Both of our kernelization algorithms run in time $\mathcal{O}\left(1.253^d \cdot (kd)^{\mathcal{O}(1)} \cdot n \log n\right)$. It is important to note that, unless the Exponential Time Hypothesis (ETH) fails, the dependence on d cannot be improved to $2^{o(d)}$, as the case $k = 0$ reduces to solving the classical Vertex Cover problem, which is known to require $2^{\Omega(d)}$ time under ETH.

A key ingredient in our kernelization algorithms is a structural result about the hereditary graph class $\mathcal{G}_d$, consisting of graphs in which every connected component has vertex cover number at most d. We show that $\mathcal{G}_d$ admits a finite obstruction set (with respect to the induced subgraph relation) of size $2^{\mathcal{O}(d^2)}$, where each obstruction graph has at most $3d + 2$ vertices. This combinatorial result may be of independent interest.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Parameterized complexity, Polynomial Kernels, Vertex Cover, Finite Forbidden Characterization

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.20

1 Introduction

Graph modification problems, especially vertex and edge deletions to achieve a desired structural property, are central to algorithmic graph theory [4, 6, 15, 30, 35]. Formally, given a fixed graph class $\mathcal{G}$, the goal is to delete at most k vertices or edges from an input graph G so that the resulting graph belongs to $\mathcal{G}$. This general framework captures a wide variety of well-studied problems, including Vertex Cover [16], Feedback Vertex Set [16], Planar Deletion [26, 28], Multiway Cut [31], Interval Deletion [9], Chordal Deletion [10], and Claw-Free Deletion [15]. A key motivation for studying such problems is that many computationally hard problems become tractable when restricted to graphs in $\mathcal{G}$ [1, 7, 15, 20]. Among these, the Vertex Cover problem is particularly fundamental. Given a graph G and an integer k, the Vertex Cover problem asks whether G contains a vertex set $S \subseteq V(G)$, also called *vertex cover* of G, of size at most k such that $G - S$ does not contain an edge.

One of the classical graph modification problems also viewed as a graph clustering problem asks whether a given graph can be transformed into a cluster graph (i.e., a graph in which every connected component is a clique) by deleting at most k vertices or k edges [23, 29]. This blends the perspectives of graph editing and clustering. Building on this line of work, and inspired by the framework proposed during the Dagstuhl Seminar 23331 [27], we consider a natural specialization of this general approach (also suggested during the seminar). The seminar introduced a broad perspective on clustering, where the goal is to partition a graph into connected components (or "clusters") that satisfy simple structural integrity constraints, not necessarily limited to cliques.

One such question proposed during the seminar is the Vertex Decomposition into Small Dominator Clusters problem. Given a graph G and parameters (k, d), the task is to determine whether at most k vertices can be deleted from G to obtain a graph G' such that each connected component of G' has domination number at most d [27]. The seminar posed the open question of whether this problem is fixed-parameter tractable (FPT) when parameterized by the combined parameters k and d. Since the problem is known to be W[2]-complete when $k = 0$ [16], it remains computationally hard even in the absence of deletions. Nonetheless, it was natural to ask whether an algorithm exists when the parameter d is allowed to appear in the exponent of n. This question was recently answered in the affirmative by Bentert et al. [5], who presented an algorithm with running time $f(k, d) \cdot n^{\mathcal{O}(d)}$.

In this work, we investigate two natural variants of this emerging family of vertex-deletion and edge-deletion clustering problems. Specifically, we study the Vertex Deletion to d-Vertex Cover Components and Edge Deletion to d-Vertex Cover Components problems, where the objective is to delete at most k vertices or edges, respectively, so that each connected component of the resulting graph has vertex cover of size at most d.

In the same seminar [27], Karypis et al. also proposed the study of the Vertex Decomposition into Small Vertex Cover Clusters problem, referred to as the Layered Vertex Cover problem. In this problem, the goal is to delete a small number of vertices so that every connected component of the resulting graph can be further reduced by deleting at most d_1 vertices, resulting in a graph where each connected component has a vertex cover of size at most d_2. This formulation is closely related to our investigation of the Vertex Deletion to d-VCC and Edge Deletion to d-VCC problems.

While in the work of Bentert et al. [5], the classical Dominating Set problem played a central role in characterizing the connected components after the deletion of a small number of vertices, in our work, the Vertex Cover problem serves an analogous foundational role.

The Vertex Cover is one of the most extensively studied problems in parameterized complexity, and has served as a cornerstone in the development of both theoretical frameworks and algorithms especially in the context of kernelization, which is a central theme of this paper [16, 19].

Kernelization is a core technique in parameterized complexity. It focuses on designing polynomial-time preprocessing algorithms that reduce a given instance to an equivalent one of size bounded by a function of the parameter, while preserving correctness guarantees (see Definition 9 for a formal definition). The Vertex Cover problem has inspired a rich toolbox of such techniques, including crown rule reductions, the Nemhauser–Trotter theorem, and LP-based kernelization methods [16, 19, 13, 32, 33, 18, 3, 2, 14, 8, 25].

1.1 Our Results and Methods

From a parameterized complexity perspective, both Vertex Deletion to d-VCC and Edge Deletion to d-VCC exhibit rich algorithmic structure. Our main result is an almost-linear-time kernelization for both problems.

▶ **Theorem 1.** *Vertex Deletion to d-VCC admits a kernel with at most $\mathcal{O}(d^6k^3)$ vertices and $\mathcal{O}(d^9k^4)$ edges. Furthermore, it can be computed in time $\mathcal{O}\left(1.253^d \cdot (kd)^{\mathcal{O}(1)} \cdot n \log n\right)$.*

▶ **Theorem 2.** *Edge Deletion to d-VCC admits a kernel with at most $\mathcal{O}(d^4k)$ vertices and $\mathcal{O}(d^5k)$ edges. Furthermore, it can be computed in time $\mathcal{O}\left(1.253^d \cdot (kd)^{\mathcal{O}(1)} \cdot n \log n\right)$.*

In the same report [27] by Dagstuhl, the importance of developing "truly linear-time" FPT algorithms was emphasized – those with running time $\mathcal{O}(n + f(k))$ – to handle large-scale network data. While recent advances have made progress in this direction [11, 12], the area remains relatively underexplored. We believe that our kernelization results also contribute meaningfully to this growing body of work.

A central ingredient in our kernelization framework is a structural result concerning the hereditary graph class $\mathcal{G}_d$, defined as the class of graphs in which every connected component has vertex cover number at most d. We show that this class admits a finite forbidden induced subgraph characterization: every minimal obstruction has at most $3d + 2$ vertices, and the total number of such obstructions is bounded by $2^{\mathcal{O}(d^2)}$. This structural result may be of independent interest and could find applications in other graph modification problems where the goal is to decompose a graph into connected components with bounded vertex cover number.

Let $\mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ denote the set of minimal forbidden induced subgraphs for $\mathcal{G}_d$. Observe that, since each graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ has at most $3d + 2$ vertices, one can design a fixed-parameter tractable (FPT) algorithm for both Vertex Deletion to d-VCC and Edge Deletion to d-VCC with running time $d^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(d)}$. The idea is to enumerate all subsets of $V(G)$ of size at most $3d+2$ and check whether any such subset induces a graph in $\mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$. If so, the algorithm branches on how to intersect the obstruction using at most k deletions. A natural question that arises is whether we can identify an obstruction graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ more efficiently than by brute-force enumeration. This leads us to the following algorithmic problem.

Membership or an Obstruction to $\mathcal{G}_d$

Input:	An undirected graph G.
Task:	Either correctly determine that $G \in \mathcal{G}_d$, or output a graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ such that $H \subseteq_i G$.

We design the following algorithm for the Membership or an Obstruction to $\mathcal{G}_d$ problem.

▶ **Theorem 3.** *Let G be a graph on n vertices. Then, in time $\mathcal{O}(1.253^d d^{\mathcal{O}(1)} n \log n)$, one can either verify that $G \in \mathcal{G}_d$, or output an induced subgraph $H \subseteq_i G$ such that $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ and $|V(H)| \leq 3d + 2$.*

It is important to note that, while the algorithm of Bentert et al. [5] necessarily incurs an $n^{\mathcal{O}(d)}$ dependence in its running time, our results achieve an almost-linear dependence on the input size. At the same time, unless the Exponential Time Hypothesis (ETH) fails, the dependence on d cannot be improved to $2^{o(d)}$ in our setting either. This is because the special case $k = 0$ reduces to the classical Vertex Cover problem, which is known to require $2^{\Omega(d)}$ time under ETH [24].

Finally, we turn to the Edge Deletion to d-VCC problem. Observe that for $d = 0$, the problem is solvable in polynomial time: in this case, the task reduces to deleting all edges from the input graph, yielding an edgeless graph, which trivially satisfies the vertex cover condition. This stands in contrast to the classical Vertex Cover problem. However, this tractability does not extend beyond $d = 0$. In fact, one can show that Edge Deletion to d-VCC becomes NP-complete already for $d = 1$. Specifically, we can prove that deciding whether one can delete k edges to obtain a graph in which every connected component is a star is NP-complete, via a reduction from the Dominating Set problem.

As a final remark, we observe that the above formulation captures a hierarchy of increasingly general problems. When $d = 0$, the vertex-deletion variant reduces to the classical Vertex Cover problem. For $d = 1$, each connected component becomes a star. As d increases, the allowed components grow in structural complexity, yet remain bounded by a core of size at most d that governs their internal interactions. This hierarchy offers a smooth trade-off between expressiveness and algorithmic tractability.

2 Preliminaries

Graphs. We use standard graph-theoretic notation, and we refer the reader to Diestel [17] for any undefined terms. We consider simple unweighted undirected graphs. For a positive integer ℓ, We use $[\ell]$ to denote the set $\{1, \ldots, \ell\}$. For a graph G, we use $V(G)$ and $E(G)$ to denote the set of vertices and edges of G, respectively. For vertices $u, v \in V(G)$, we write uv to indicate that $\{u, v\} \in E(G)$. For a graph G and a vertex set $S \subseteq V(G)$, we define $G - S$ to be the graph obtained after deleting the set of vertices S from G. If $S \subseteq V(G)$ is singleton, that is $S = \{s\}$, then we use $G - s$ instead of $G - \{s\}$. Similarly, for a subset of edges $S \subseteq E(G)$, we define $G - S$ to be the graph obtained after deleting the set of edges S from G. If $S \subseteq E(G)$ is a singleton, that is $S = \{uv\}$, then we use $G - uv$ instead of $G - \{uv\}$.

For a vertex $v \in V(G)$, we say that u is a *neighbor* of v if uv is an edge in G. Further, we denote *neighborhood* of a vertex v in graph G as $\mathsf{N}_G(v)$ to be the set of vertices that are neighbor to v, i.e., we have $\mathsf{N}_G(v) := \{u \mid uv \in E(G)\}$. For a set of vertices $S \subseteq V(G)$, the neighborhood of S is denoted by $\mathsf{N}_G(S)$ and defined as $\mathsf{N}_G(S) := \left(\bigcup_{v \in S} \mathsf{N}_G(v)\right) \setminus S$. Similarly, for a set of vertices $S \subseteq V(G)$, the closed neighborhood of S is denoted by $\mathsf{N}_G[S]$ and defined as $\mathsf{N}_G[S] := \left(\bigcup_{v \in S} \mathsf{N}_G(v)\right) \cup S$. We omit the subscript G when the graph is clear from the context. For a graph G and a vertex v, we denote by E_v the set of edges incident to v, that is, $E_v := \{uv \mid uv \in E(G)\}$.

We say that a graph H is a subgraph of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Further we say that a graph H is an induced subgraph of G if $V(H) \subseteq V(G)$ and $E(H) = \{uv \mid \{u, v\} \subseteq V(H) \text{ and } uv \in E(G)\}$. We define the neighborhood of H as $\mathsf{N}_G(V(H)) \setminus V(H)$ and denote

it by $\mathsf{N}_G(H)$. We say that an edge $uv \in E(G)$ is an *internal edge* of H if $u, v \in V(H)$. For a pair of vertices $u, v \in V(G)$, a $u - v$ path is a sequence of vertices $(u = x_1, x_2, \ldots, x_k = v)$ such that $x_i x_{i+1} \in E(G)$ for all $i \in [k-1]$. The vertices $x_2, \ldots, x_{k-1}$ are called the *internal vertices* of the $u - v$ path. The *length* of a path is defined as the number of vertices it contains.

The *distance* between two vertices u and v in a graph G, denoted by $\text{dist}_G(u, v)$, is defined as the length of any shortest path between them in G. A subgraph T of G is called a tree if it is connected and acyclic. We say that T is a rooted tree if there is a designated root, and we denote the root of T as $root(T)$.

We denote $\mathtt{vc}(G)$ to be the size of a minimum vertex cover of G, and alternately we call it as *vertex cover number* of G and $\mathtt{minvc}(G)$ to be a minimum vertex cover set of G. Further, we denote by $\mathcal{CC}(G)$ the set of connected components in G. Note that every connected component of a graph G is an induced subgraph of G. For a connected component $C \in \mathcal{CC}(G)$, we define $\mathtt{vc}(C)$ to be the size of a minimum vertex cover of the connected component C of G.

Further, we define the *boundary* of a set of vertices in G as follows:

▶ **Definition 4** (Boundary of a set of vertices [19]). *For a graph G and a vertex subset $S \subseteq V(G)$, we define the* boundary *of S as*

$$\partial_G(S) = \{v \in S \mid N_G(v) \setminus S \neq \emptyset\}.$$

We refer to $\partial_G(S)$ as the boundary of S. When the graph G is clear from context, we omit the subscript and write $\partial(S)$.

Now we state the well-known Hall's theorem which we need in Section 3.

▶ **Theorem 5** (Hall's theorem [17]). *Let $G = (A \cup B, E)$ be a bipartite graph with bipartition A and B. Then, there exists a matching that matches every vertex in A to a distinct vertex in B if and only if for every subset $S \subseteq A$, it holds that $|N_G(S)| \geq |S|$.*

▶ **Theorem 6** (Hopcroft-Karp algorithm [22, 16]). *Let G be an undirected bipartite graph with bipartition V_1 and V_2, on n vertices and m edges. Then we can find a maximum matching as well as a minimum vertex cover of G in time $\mathcal{O}(m\sqrt{n})$. Furthermore, in time $\mathcal{O}(m\sqrt{n})$, either we can find a matching saturating V_1, or an inclusion-wise minimal set $X \subseteq V_1$ such that $|N(X)| < |X|$.*

▶ **Definition 7** (Tree Decomposition [16]). *A* tree decomposition *of a graph $G = (V, E)$ is a pair $(\mathcal{T}, \{X_t \subseteq V \mid t \in V(\mathcal{T})\})$, where $\mathcal{T}$ is a tree and each node t of $\mathcal{T}$ is assigned a set $X_t \subseteq V$, called a* bag*, such that:*

1. *For every vertex $v \in V$, there exists at least one bag X_t such that $v \in X_t$.*
2. *For every edge $\{u, v\} \in E$, there exists a bag X_t such that $\{u, v\} \subseteq X_t$.*
3. *For every vertex $v \in V$, the set of nodes $\{t \in V(\mathcal{T}) \mid v \in X_t\}$ induces a connected subtree of $\mathcal{T}$.*

▶ **Definition 8** (Tree Width [16]). *The* width *of a tree decomposition is the size of its largest bag minus one, i.e., $\max_{t \in V(\mathcal{T})} |X_t| - 1$. The* treewidth *of G, denoted $\mathsf{tw}(G)$, is the minimum width over all possible tree decompositions of G.*

It is easy to note that for a graph G, we have $\mathsf{tw}(G) \leq \mathtt{vc}(G)$, where $\mathtt{vc}(G)$ is the size of a minimum vertex cover of G [16].

Let $\mathtt{vc}_c(G)$ denote the maximum over vertex cover number of all the connected components of G, that is,

$$\mathtt{vc}_c(G) \coloneqq \max\{\mathtt{vc}(G[C]) \mid C \text{ is a connected component of } G\}.$$

We say that a graph G satisfies $\mathtt{vc}_c(G) \leq d$ if every connected component of G has vertex cover number at most d.

Due to space constraints, the proofs of the results marked (★) will be presented in the full version of the paper.

Parameterized algorithms and Kernelization. In parameterized complexity, computational problems are analyzed based on two inputs: the main input size n and an additional parameter k, which typically captures some structural aspect of the instance.

Kernelization is a formal framework for polynomial-time preprocessing in parameterized complexity.

▶ **Definition 9** (Kernelization). *Given an instance (I, k) of a parameterized problem, a kernelization algorithm transforms it in polynomial time into a new instance (I', k') – called the* kernel *– such that:*

- *the size of (I', k') is bounded by a function $f(k)$ (independent of the original input size), and*
- *the new instance is equivalent to the original one; that is, (I, k) is a YES-instance if and only if (I', k') is.*

If $f(k)$ is a polynomial function, then the kernel is referred to as a polynomial kernel.

For a detailed introduction to parameterized algorithms and kernelization, we refer the reader to the textbooks [16, 19].

3 Bounding the Size of Minimal Obstructions and an Algorithm to Compute it

For a non-negative integer d, let $\mathcal{G}_d$ denote the class of graphs in which every connected component has vertex cover number at most d. That is,

$$\mathcal{G}_d = \{G \mid \mathtt{vc}_c(G) \leq d\}.$$

Note that $\mathcal{G}_d$ is a *hereditary* graph class – meaning that if $G \in \mathcal{G}_d$, then every induced subgraph of G also belongs to $\mathcal{G}_d$.

Our problems, Vertex Deletion to d-VCC and Edge Deletion to d-VCC, correspond to deleting at most k vertices or k edges, respectively, to obtain a graph in $\mathcal{G}_d$. A standard approach for designing FPT algorithms or polynomial kernels is to identify a *finite forbidden subgraph characterization* of the base class $\mathcal{G}_d$. In this section, we pursue this approach. For a fixed integer d, we also develop a recognition algorithm that, given a graph G, runs in near-linear time and either confirms that $G \in \mathcal{G}_d$, or returns a small induced subgraph that certifies $G \notin \mathcal{G}_d$ (see Theorem 3).

3.1 Finite Forbidden Characterization

To show that $\mathcal{G}_d$ admits a finite forbidden characterization under the induced subgraph relation, we begin by defining the notion of minimal obstructions for hereditary graph classes. A graph H is called an *obstruction* for a graph class $\mathcal{F}$ if $H \notin \mathcal{F}$. We now formally define the family of minimal such obstructions.

▶ **Definition 10** (Family of Minimal Obstructions). *Let $\mathcal{F}$ be a hereditary graph class. A graph H is a* minimal obstruction *for $\mathcal{F}$ under the induced subgraph relation if $H \notin \mathcal{F}$, but every proper induced subgraph of H belongs to $\mathcal{F}$. The set of all such minimal obstructions is denoted by $\mathsf{Obs}_{\subseteq_i}(\mathcal{F})$, and is defined as:*

$$\mathsf{Obs}_{\subseteq_i}(\mathcal{F}) = \{H \notin \mathcal{F} \mid \text{for all } H' \subsetneq_i H,\ H' \in \mathcal{F}\}.$$

Here, $H' \subsetneq_i H$ indicates that H' is a proper induced subgraph of H.

The following lemma establishes that all minimal obstructions for the class $\mathcal{G}_d$ are necessarily connected graphs.

▶ **Lemma 11.** *Let $\mathcal{G}_d$ be the graph class defined above. Then every graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ is connected.*

Proof. Suppose, for the sake of contradiction, that there exists a graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ that is disconnected and has at least two connected components. Since $\mathtt{vc}_c(H) > d$, at least one connected component, say C, must have vertex cover number greater than d. But then $H[V(C)]$ is a proper induced subgraph of H and also violates the condition $\mathtt{vc}_c(G) \leq d$, implying $H[V(C)] \notin \mathcal{G}_d$. This contradicts the minimality of H, as not all its proper induced subgraphs belong to $\mathcal{G}_d$. Hence, every minimal obstruction must be connected. ◀

Next, we establish several structural properties of the minimal obstructions for $\mathcal{G}_d$. These results will later be combined in Lemma 16 to show that the set of minimal obstructions for $\mathcal{G}_d$ is finite and its size is bounded by $2^{\mathcal{O}(d^2)}$. Moreover, we show that every such obstruction has a number of vertices linear in d. As a first step, we prove that every minimal obstruction for $\mathcal{G}_d$ has vertex cover number exactly $d+1$.

▶ **Lemma 12.** *Let $\mathcal{G}_d$ be defined as above. Let $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ then $\mathtt{vc}(H) = d+1$.*

Proof. By Definition 10, since $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$, we know that $H \notin \mathcal{G}_d$. This implies that $\mathtt{vc}(H) \geq d+1$. To complete the proof, we show that $\mathtt{vc}(H) \leq d+1$. Suppose, for contradiction, that $\mathtt{vc}(H) \geq d+2$.

Let T be a spanning tree of H, and let x be a leaf of T. Then the graph $H' = H \setminus \{x\}$ is connected, since removing a leaf from a spanning tree does not disconnect it. Moreover, $H' \subsetneq_i H$. Observe that removing a vertex from a graph can decrease the size of a minimum vertex cover by at most one. Therefore,

$$\mathtt{vc}(H') \geq \mathtt{vc}(H) - 1 \geq (d+2) - 1 = d+1.$$

This implies that $H' \notin \mathcal{G}_d$, contradicting the fact that H is a minimal obstruction – since all proper induced subgraphs of H must belong to $\mathcal{G}_d$.

We conclude that our assumption $\mathtt{vc}(H) \geq d+2$ is false. Hence, $\mathtt{vc}(H) = d+1$, as claimed. ◀

To establish a bound on the size of each graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$, we make use of the following well-known result, which relates the vertex cover number and the connected vertex cover number of a connected graph. While the size bound is well known, we are not aware of any existing implementation that computes the corresponding set efficiently in linear time.

Proofs of the lemmata marked with ★ are provided in the full version of the paper.

▶ **Lemma 13 (★).** *Let G be a connected graph, and let $S \subseteq V(G)$ be a vertex cover of G. Then there exists a set $Y \subseteq V(G) \setminus S$ with $|Y| \leq |S| - 1$ such that the induced subgraph $G[S \cup Y]$ is connected. Moreover, such a set Y can be computed in $\mathcal{O}(m+n)$ time.*

Next, our goal is to establish an upper bound on the size of each minimal obstruction in $\mathcal{G}_d$. To achieve this, we first prove the following lemma, which we use in the subsequent lemma to derive the desired size bound on minimal obstructions.

▶ **Lemma 14 (★).** *Let G be a connected graph with $\mathtt{vc}(G) = \ell$. Suppose we are given sets $S \subseteq S^\star$, where S is a vertex cover of size ℓ and $G[S^\star]$ is connected. Let $I = V(G) \setminus S^\star$, and suppose $I' \subseteq I$ with $|I'| = \ell + 1$. Then there exists a vertex $v \in I'$ such that $G - v$ remains connected and $\mathtt{vc}(G - v) = \ell$. Moreover, such a vertex can be found in time $\mathcal{O}(\ell^{2.5})$.*

We are now ready to prove the upper bound on the size of an obstruction.

▶ **Lemma 15.** *Let $\mathcal{G}_d$ be as defined above. Let $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ then $|V(H)| \leq 3d + 2$.*

Proof. Suppose for contradiction that there exists a graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ such that $|V(H)| \geq 3d + 3$. By the definition of $\mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$, we have that H is connected and $\mathtt{vc}(H) = d + 1$.

Let S be a minimum vertex cover of H, so $|S| = d + 1$. The complement $I := V(H) \setminus S$ is an independent set, and we have:

$$|I| = |V(H)| - |S| \geq (3d + 3) - (d + 1) = 2d + 2.$$

Since H is connected and S is a vertex cover, we can apply Lemma 13 to find a subset $Y \subseteq I$ of size at most d such that the subgraph $H[S \cup Y]$ is connected. Let $S^\star = S \cup Y$, and observe that $G[S^\star]$ is connected and $|S^\star| \leq d + 1 + d = 2d + 1$.

Now define $I^\star := I \setminus Y$. Then

$$|I^\star| \geq |I| - |Y| \geq (2d + 2) - d = d + 2.$$

Now, let $I' \subseteq I^\star$ be an arbitrary subset of size exactly $\ell + 1$ where $\ell = d + 1 = \mathtt{vc}(H)$. Thus, all conditions of Lemma 14 are satisfied: H is connected, $\mathtt{vc}(H) = \ell$, S is a vertex cover of size ℓ, $S \subseteq S^\star$ with $H[S^\star]$ connected, and $I' \subseteq V(H) \setminus S^\star$ with $|I'| = \ell + 1$.

By Lemma 14, there exists a vertex $v \in I'$ such that the graph $H' := H - v$ is connected and $\mathtt{vc}(H') = \ell = d+1$. But then H' is a proper induced subgraph of H that is still connected and has vertex cover number $d + 1$. This contradicts the assumption that $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$, since H would not be minimal.

Thus, our assumption was false and we must have $|V(H)| \leq 3d + 2$, completing the proof. ◀

We combine the results obtained above in this section to prove the following lemma.

▶ **Lemma 16.** *Let $\mathcal{G}_d$ be as defined above. Then the obstruction set $\mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ contains at most $2^{\mathcal{O}(d^2)}$ graphs. Moreover, every graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ has at most $3d + 2$ vertices.*

Proof. By Lemma 15, every graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ has at most $3d + 2$ vertices and vertex cover number $\mathtt{vc}(H) = d + 1$. Therefore, the total number of such obstructions is bounded by the number of connected graphs on at most $3d + 2$ vertices. This number is at most $2^{(3d+2)^2} = 2^{\mathcal{O}(d^2)}$, yielding the claimed bound on the size of the obstruction set. ◀

3.2 Membership or an Obstruction to $\mathcal{G}_d$

In this section, we present an algorithm for Membership or an Obstruction to $\mathcal{G}_d$ parameterized by d. In particular, we prove the following result.

▶ **Theorem 3.** *Let G be a graph on n vertices. Then, in time $\mathcal{O}(1.253^d d^{\mathcal{O}(1)} n \log n)$, one can either verify that $G \in \mathcal{G}_d$, or output an induced subgraph $H \subseteq_i G$ such that $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ and $|V(H)| \leq 3d + 2$.*

For our algorithm, we require the following result, which combines a linear-time kernel for the classical Vertex Cover problem [34] with the currently known fastest parameterized algorithm for Vertex Cover [21].

▶ **Proposition 17.** *Vertex Cover can be solved in time* $\mathcal{O}(1.253^{\eta} \cdot \eta^{\mathcal{O}(1)} + \eta n)$ *or* $\mathcal{O}(2^{\eta} \cdot \eta^3 + \eta n)$ *on an instance* (G, η)*, where* $|V(G)| = n$*.*

We use this result as a subroutine in our algorithm for Membership or an Obstruction to $\mathcal{G}_d$, where we need to repeatedly solve Vertex Cover on various induced subgraphs of the input graph. Since the parameter d in Membership or an Obstruction to $\mathcal{G}_d$ bounds the vertex cover number of each connected component in the target graph, we can apply the proposition with $\eta \leq d+1$, ensuring that each call to Vertex Cover runs efficiently with respect to the parameter d.

Our proof proceeds in the following steps:

Step 0. We begin by applying a modified breadth-first search (BFS) that either determines that the total number of edges in G is at most dn, or returns a subgraph $G' \subseteq G$ with $|E(G')| = dn + 1$.
Observe that if $G \in \mathcal{G}_d$, then the number of edges in G is upper bounded by dn. Indeed, if a connected component C has vertex cover number at most d, then the number of edges in C is at most $d \cdot |V(C)|$. Summing this bound over all connected components yields the claimed global bound of dn on the total number of edges.
In the latter case, when the algorithm returns a subgraph $G' \subseteq G$ with $dn+1$ edges, we can immediately proceed to Step 2. This entire step can be executed in time $\mathcal{O}(dn)$.

Step 1. Given a graph G, we first use Proposition 17 to check whether $\mathtt{vc}_c(G) \leq d$. If this test fails, then there exists a connected component $C \subseteq G$ such that $\mathtt{vc}(G[C]) > d$. From this point onward, we focus on the subgraph $G[C]$, and aim to output an obstruction $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ with $V(H) \subseteq V(C)$. Without loss of generality, we refer to $G[C]$ as G itself, as we may assume G is connected.

Step 2. In the second step, we identify a connected induced subgraph $H'' \subseteq G$ such that $\mathtt{vc}(H'') = d+1$.

Step 3. Next, we obtain a subgraph $H' \subseteq H''$ by making the argument of Lemma 15 constructive. This guarantees that $|V(H')| \leq 3d+1$. However, H' may not yet belong to $\mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$.

Step 4. Finally, by removing additional vertices from H', we obtain a graph $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$.

This completes the description of the algorithm for Membership or an Obstruction to $\mathcal{G}_d$. Next, we show the implementation of some of the non-trivial steps below. The implementation of Steps 2, 3 and 4 will be provided in the full version of the paper.

Running time. The initial graph H' has at most $3d+2$ vertices. In each iteration, we go over all vertices $v \in V(H)$ and compute the vertex cover number of $H - v$. This takes $\mathcal{O}(3d+2)$ vertex cover computations per step as the number of connected component in $H-v$ is $\mathcal{O}(3d+2)$. Since one vertex is removed in each iteration, the total number of steps is at most $3d+2$. Hence, the total running time to implement this step is bounded by $1.253^d \cdot d^{\mathcal{O}(1)}$.

4 Cubic Vertex Kernel for Vertex Deletion to d-VCC

This section presents a polynomial kernel for Vertex Deletion to d-VCC parameterized by k. As a first step, we compute an $\mathcal{O}(d)$-approximate solution for Vertex Deletion to d-VCC, which serves as the basis for further reduction.

4.1 Approximation Algorithm for Vertex Deletion to d-VCC

▶ **Lemma 18.** *Let (G, k) be an instance of* Vertex Deletion to d-VCC. *Then, in time $\mathcal{O}(k \cdot (1.253^d d^{\mathcal{O}(1)} n \log n))$, we can either find a set $S \subseteq V(G)$ of size at most $(3d+2)k$ such that $\mathtt{vc}_c(G - S) \leq d$, or correctly conclude that (G, k) is a* No*-instance of* Vertex Deletion to d-VCC.

Proof. We iteratively construct a solution $S \subseteq V(G)$ of size at most $k(3d+2)$ by repeatedly identifying a minimal obstruction using Theorem 3. We also maintain a family $\mathcal{P}$ of induced subgraphs and a counter $k^\star \leftarrow 0$. The procedure proceeds as follows:

1. While G contains a minimal obstruction $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ and $k^\star \leq k+1$, do:
   a. Add $V(H)$ to the solution: $S \leftarrow S \cup V(H)$,
   b. Add H to the family: $\mathcal{P} \leftarrow \mathcal{P} \cup \{H\}$,
   c. Increment the counter: $k^\star \leftarrow k^\star + 1$,
   d. Remove $V(H)$ from G.
2. If $k^\star = k+1$, return that (G, k) is a No-instance of Vertex Deletion to d-VCC.
3. Otherwise, return the set S.

Correctness follows from the fact that the obstructions added to $\mathcal{P}$ are pairwise vertex-disjoint. Since each obstruction is a minimal forbidden induced subgraph for $\mathcal{G}_d$, any feasible solution must delete at least one vertex from each obstruction to ensure that the resulting graph belongs to $\mathcal{G}_d$. Thus, if we identify $k+1$ such disjoint obstructions, any solution must contain at least $k+1$ vertices, implying that (G, k) is a No-instance.

In the other case, we remove at most k disjoint obstructions. By Lemma 15, each obstruction contains at most $3d+2$ vertices. Hence, the total number of vertices added to S is at most $k(3d+2)$. Moreover, since $G - S$ contains no induced obstruction, we have $\mathtt{vc}_c(G - S) \leq d$, and therefore $G - S \in \mathcal{G}_d$.

Finally, since we invoke Theorem 3 at most $k+1$ times, the overall running time is bounded accordingly. This concludes the proof. ◀

4.2 Marking and Irrelevant Vertices

From this point onward, we assume that we are given a set $S \subseteq V(G)$ of size at most $(3d+2)k$ such that $\mathtt{vc}_c(G - S) \leq d$. We now examine the connected components of the graph $G - S$. Let $\mathcal{CC}(G - S)$ denote the set of connected components of $G - S$. We partition $\mathcal{CC}(G - S)$ into two subsets based on the size of the components: We now partition the set of connected components $\mathcal{CC}(G - S)$ based on their size:

- Let $\mathcal{C}_1 := \{C \in \mathcal{CC}(G - S) \mid |V(C)| = 1\}$ denote the set of singleton components (i.e., isolated vertices).
- Let $\mathcal{C}_2 := \{C \in \mathcal{CC}(G - S) \mid |V(C)| > 1\}$ denote the set of components with more than one vertex.

We first introduce the following simple reduction rule, which removes all connected components C of G such that $\mathtt{vc}(C) \leq d$. The safeness of this rule is immediate.

▶ **Reduction Rule 1.** *Let $C \in \mathcal{CC}(G - S)$ be a connected component such that every vertex $v \in V(C)$ satisfies $N_G(v) \cap S = \emptyset$. In this case, remove C from the graph. The resulting instance is $(G - C, k)$.*

Next, we aim to bound the number of connected components in $\mathcal{CC}(G - S)$ that contain more than one vertex, i.e., the components in the set $\mathcal{C}_2$. To achieve this, we make use of the Expansion Lemma.

▶ **Definition 19** (q-Expansion). *Let G be a bipartite graph with vertex bipartition (A, B), and let $X \subseteq A$, $Y \subseteq B$. We say that X has a* q-expansion into Y *if there exists a collection of $q|X|$ edges from X to Y such that:*

- *Each vertex in X is incident to exactly q of these edges, and*
- *The endpoints of these $q|X|$ edges in Y are distinct (i.e., no vertex in Y is incident to more than one of the selected edges).*

Equivalently, there exists a collection of q pairwise vertex-disjoint matchings from X into Y.

▶ **Lemma 20** (Expansion Lemma [19]). *Let q be a positive integer, and let G be a bipartite graph with vertex bipartition (A, B) such that:*

(i) *$|B| \geq q|A|$, and*

(ii) *There are no isolated vertices in B.*

Then, there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that:

- *The set X has a q-expansion into Y, and*
- *No vertex in Y has a neighbor outside X, that is, $\mathsf{N}(Y) \subseteq X$.*

Furthermore, the sets X and Y can be found in time $\mathcal{O}(m\sqrt{n})$.

To bound the number of non-trivial connected components in $G - S$, we apply the *q-Expansion Lemma*. Specifically, we aim to upper-bound the number of components in $\mathcal{C}_2$. To this end, we construct a bipartite graph $B = (S \cup \mathcal{C}_2, E_B)$ that captures the adjacency between the approximate deletion set S and the components in $\mathcal{C}_2$. Formally, the bipartition of B is defined as follows:

- The left side of the bipartition corresponds to the set $S \subseteq V(G)$, and
- The right side contains one representative vertex for each connected component $C \in \mathcal{C}_2$.

We add an edge between a vertex $p \in S$ and a component $C \in \mathcal{C}_2$ if and only if p is adjacent (in G) to at least one vertex in C; that is, if $N_G(p) \cap V(C) \neq \emptyset$. We slightly abuse notation by referring to both a connected component $C \in \mathcal{C}_2$ and its corresponding vertex in B using the same symbol.

Note that by Reduction Rule 1, every component in $\mathcal{C}_2$ has at least one neighbor in S; hence, there are no isolated vertices on the right side of the bipartite graph B. We now apply the *Expansion Lemma* (Lemma 20) to this bipartite graph to argue that if $|\mathcal{C}_2|$ is too large relative to $|S|$, then the instance must be a `No`-instance. This leads to the following reduction rule:

▶ **Reduction Rule 2.** *If $|\mathcal{C}_2| \geq (d+1)|S|$, then apply the algorithm guaranteed by the* Expansion Lemma *to compute sets $X \subseteq S$ and $Y \subseteq \mathcal{C}_2$ such that X has a $(d+1)$-expansion into Y and $N_B(Y) \subseteq X$. Remove the vertices in X from G and reduce the parameter accordingly. The resulting instance is $(G - X, k - |X|)$.*

To establish the correctness of Reduction Rule 2, we prove the following lemma.

▶ **Lemma 21 (★).** *Reduction Rule 2 is safe; that is, the instance (G, k) of* Vertex Deletion to d-VCC *is a `Yes`-instance if and only if the reduced instance $(G - X, k - |X|)$ is a `Yes`-instance.*

After exhaustively applying Reduction Rules 1 and 2, we obtain that the number of non-trivial components in $\mathcal{C}_2$ satisfies the bound $|\mathcal{C}_2| \leq (d+1)(3d+2)k$. However, if we remove these connected components one by one as suggested, the overall running time may no longer remain bounded by $\mathcal{O}((kd)^{\mathcal{O}(1)} n \log n)$.

To address this, we proceed as follows. Let $Q \subseteq S$ be the subset of vertices in S that have at least $k + d + 1$ neighbors on the right side of the bipartite graph B, i.e., into $\mathcal{C}_2$. We claim that every solution $Z \subseteq V(G)$ of size at most k must contain all vertices in Q.

Indeed, suppose for contradiction that there exists a vertex $v \in Q \setminus Z$. Since v has at least $k+d+1$ neighbors in $\mathcal{C}_2$, and Z contains at most k vertices, there are at least $d+1$ components in $\mathcal{C}_2$ that are completely disjoint from Z. But then, in the graph $G-Z$, the vertex v, along with these $d+1$ components, induces a subgraph that requires a vertex cover of size at least $d+1$, contradicting the assumption that Z is a valid solution (i.e., $\mathtt{vc}_c(G-Z) \leq d$). This proves that all vertices in Q must belong to any valid solution of size at most k.

▶ **Reduction Rule 3.** *Let $Q \subseteq S$ be the subset of vertices in S that have at least $k+d+1$ neighbors on the right side of the bipartite graph B, i.e., into $\mathcal{C}_2$. The new instance is $(G-Q, k-|Q|)$.*

Since the number of edges in the bipartite graph B is upper bounded by the number of edges in G, we can compute the set Q in time $\mathcal{O}(n(d+k))$. After removing Q, let $\mathcal{C}_0 \subseteq \mathcal{C}_2$ denote the set of components that become isolated on the right side of B. These correspond to all connected components $C \subseteq G-Q$ such that $\mathtt{vc}(C) \leq d$. By applying Reduction Rule 1, we can remove all such components in $\mathcal{C}_0$ simultaneously. This step also takes linear time. Following this reduction, the number of remaining non-trivial components – i.e., the size of $\mathcal{C}_2$, which corresponds to the non-isolated vertices on the right side of the updated bipartite graph B – is upper bounded by $(3d+2)k(k+d)$. We now apply Reduction Rule 2 exhaustively on this reduced bipartite graph to further shrink $\mathcal{C}_2$ to at most $(3d+2)kd$ components. The number of times Reduction Rule 2 needs to be applied is at most $\mathcal{O}(k^2d)$, and each invocation takes time $\mathcal{O}(|E(B)| \cdot \sqrt{|V(B)|})$. Since the number of edges in B is at most $\mathcal{O}(d^2k^3)$ and the number of vertices is at most $\mathcal{O}(k^2d)$, the total time required for this step is bounded by $\mathcal{O}(k^6d^{3.5})$. We summarize our results in the following lemma.

▶ **Lemma 22.** *Given an instance (G,k) of Vertex Deletion to d-VCC and a set $S \subseteq V(G)$ of size at most $(3d+2)k$ such that $\mathtt{vc}_c(G-S) \leq d$, we can compute, in time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$, an equivalent instance (G',k') of Vertex Deletion to d-VCC, where G' is an induced subgraph of G, satisfying the following properties:*

- *The graph G' has no connected component C such that $\mathtt{vc}(C) \leq d$, and*
- *The number of non-trivial connected components (i.e., $\mathcal{C}_2$) in $G'-S$ is at most $(3d+2)kd$.*

Let (G',k') be the equivalent instance of Vertex Deletion to d-VCC obtained from (G,k) by application of Lemma 22. For clarity of presentation, we rename (G',k') back to (G,k). We now enhance the set S to a new set $S' \subseteq V(G)$ such that $G-S'$ is an independent set; in other words, S' is a vertex cover of G. To this end, observe that each connected component in $\mathcal{C}_2$ has a vertex cover of size at most d, and by previous reductions, the number of such components is at most $(3d+2)kd$. In particular, we get the following.

▶ **Lemma 23 (★).** *Let (G,k) be an instance of Vertex Deletion to d-VCC where G has no connected component with vertex cover at most d, and let $S \subseteq V(G)$ be a set of size at most $(3d+2)k$ such that $\mathtt{vc}_c(G-S) \leq d$. Then, in $\mathcal{O}(1.253^d \cdot d^{\mathcal{O}(1)} \cdot n)$ time, we can compute a set $S' \supseteq S$ such that:*

- *S' is a vertex cover of G, and*
- *$|S'| \leq (3d+2)k + d \cdot (3d+2)kd = \mathcal{O}(d^3k)$.*

Next, we apply a marking scheme to bound the size of the independent set $G-S'$ obtained after augmenting S.

▶ **Marking Scheme 1.** *The marking procedure is defined as follows:*

(1) *For each $p \in S'$, mark $d+k+1$ neighbors of p in $G-S'$.*

(2) *For each pair $p_i, p_j \in S'$, mark $k+1$ common neighbors of p_i and p_j in $G-S'$.*

Let Mark be the set of marked vertices. After marking the vertices of $G - S'$ we give the following reduction rule.

▶ **Reduction Rule 4.** *Let $U = V(G) \setminus (S' \cup \mathsf{Mark})$ be the set of unmarked vertices of $G - S'$. The new instance is $(G - U, k)$.*

To prove the safeness of above Reduction Rule 4, we proof the following lemma.

▶ **Lemma 24 (★).** *The instance (G, k) of Vertex Deletion to d-VCC is a* Yes*-instance if and only if $(G - U, k)$ is a* Yes*-instance of Vertex Deletion to d-VCC.*

We now describe an implementation of Reduction Rule 4.

For every pair of vertices $(u, v) \in S' \times S'$, we maintain a set $\mathsf{Mark}(u, v)$, where u may be equal to v. Along with each such pair, we keep a counter $k_{(u,v)}$ that tracks the size of $\mathsf{Mark}(u, v)$. Initially, for all $(u, v) \in S' \times S'$, we set $\mathsf{Mark}(u, v) = \emptyset$ and $k_{(u,v)} = 0$.

Next, we iterate over each vertex $x \in V(G) \setminus S'$ and perform the following operations:

- For every neighbor $y \in \mathsf{N}(x)$, if $k_{(y,y)} \leq k + d$, then add x to $\mathsf{Mark}(y, y)$ and increment the counter: $k_{(y,y)} \leftarrow k_{(y,y)} + 1$.
- For every unordered pair $y, z \in \mathsf{N}(x)$ with $y \neq z$, if $k_{(y,z)} \leq k$, then add x to $\mathsf{Mark}(y, z)$ and increment the counter: $k_{(y,z)} \leftarrow k_{(y,z)} + 1$.

Note that since $V(G) \setminus S'$ is an independent set, each vertex $x \in V(G) \setminus S'$ has all of its neighbors in S'. By earlier bounds, we know that $|S'| = \mathcal{O}(d^3k)$, and hence $|\mathsf{N}(x)| \leq \mathcal{O}(d^3k)$. Therefore, for each vertex x, we perform at most $\mathcal{O}(|\mathsf{N}(x)|^2) = \mathcal{O}(d^6k^2)$ operations.

As a result, the entire marking procedure (Marking Scheme 1) and application of Reduction Rule 4 can be carried out in total time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$.

▶ **Lemma 25.** *Reduction Rule 4 can be applied in time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$.*

The size of the kernel is upper bounded by the number of vertices in S' and those in the marked sets. Specifically, the total number of vertices is bounded by:

$$\mathcal{O}\left(d^3k + d^6k^2(k+1) + d^3k(k+d+1)\right) = \mathcal{O}(d^6k^3).$$

Furthermore, if the number of edges in the reduced graph exceeds $\mathcal{O}(d^9k^4)$, we can safely return a trivial No-instance, as such a graph cannot admit a solution of size at most k within the class $\mathcal{G}_d$. The running time of the overall algorithm is computed as follows:

- **Approximate solution:** Compute the set S, an approximate solution, using Lemma 18, in time $\mathcal{O}(k \cdot (1.253^d d^{\mathcal{O}(1)} n \log n))$.
- **Vertex cover augmentation:** Compute a set $S' \supseteq S$ such that:
  - S' is a vertex cover of G, and
  - $|S'| \leq (3d+2)k + d \cdot (3d+2)kd = \mathcal{O}(d^3k)$,

  using Lemmas 22 and 23, in time $\mathcal{O}(1.253^d \cdot d^{\mathcal{O}(1)} \cdot n + (kd)^{\mathcal{O}(1)}n)$.
- **Applying Reduction Rule 4:** This is done using Lemma 25, in time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$.

Hence, the overall running time of the algorithm is

$$\mathcal{O}(1.253^d \cdot (kd)^{\mathcal{O}(1)} \cdot n \log n).$$

This gives us the following result.

▶ **Theorem 1.** *Vertex Deletion to d-VCC admits a kernel with at most $\mathcal{O}(d^6k^3)$ vertices and $\mathcal{O}(d^9k^4)$ edges. Furthermore, it can be computed in time $\mathcal{O}\left(1.253^d \cdot (kd)^{\mathcal{O}(1)} \cdot n \log n\right)$.*

5 Linear Vertex Kernel for Edge Deletion to d-Vertex Cover Components

In this section, we design a polynomial kernel for Edge Deletion to d-VCC. Our approach follows the same high-level outline as the one used in Section 4. As a first step, we compute an $\mathcal{O}(d^2)$-approximate solution for Edge Deletion to d-VCC.

5.1 Approximation Algorithm for Edge Deletion to d-VCC

▶ **Lemma 26.** *Let (G, k) be an instance of Edge Deletion to d-VCC. Then, in time $\mathcal{O}(k \cdot (1.253^d d^{\mathcal{O}(1)} n \log n))$, we can either find a set $P \subseteq E(G)$ of size $\mathcal{O}(d^2 k)$ such that $\mathtt{vc}_c(G - S) \leq d$, or correctly conclude that (G, k) is a* No*-instance of Edge Deletion to d-VCC.*

Proof. We iteratively construct a solution $P \subseteq E(G)$ of size $\mathcal{O}(d^2 k)$ by repeatedly enumerating edge-disjoint obstructions of bounded size (in number of edges). This is accomplished by identifying a minimal obstruction for induced subgraph relation using Theorem 3. Note that this may not be minimal obstruction with respect to subgraph relation. From Lemma 15, the number of vertices in the obstruction is upper bounded by $3d + 2$, hence the number of edges by $(9d^2 + 12d + 4)$. We also maintain a family $\mathcal{P}$ of induced subgraphs and a counter $k^\star \leftarrow 0$. The procedure proceeds as follows:

1. While G contains a minimal obstruction $H \in \mathsf{Obs}_{\subseteq_i}(\mathcal{G}_d)$ and $k^\star \leq k + 1$, do:
   a. Add $E(H)$ to the solution: $P \leftarrow P \cup E(H)$,
   b. Add H to the family: $\mathcal{P} \leftarrow \mathcal{P} \cup \{H\}$,
   c. Increment the counter: $k^\star \leftarrow k^\star + 1$,
   d. Remove $E(H)$ from G.
2. If $k^\star = k + 1$, return that (G, k) is a No-instance of Edge Deletion to d-VCC.
3. Otherwise, return the set P.

Correctness follows from the fact that the obstructions added to $\mathcal{P}$ are pairwise edge-disjoint. Since each obstruction is a minimal forbidden induced subgraph for $\mathcal{G}_d$, any feasible solution must delete at least one edge from each obstruction to ensure that the resulting graph belongs to $\mathcal{G}_d$. Thus, if we identify $k + 1$ such disjoint obstructions, any solution must contain at least $k + 1$ edges, implying that (G, k) is a No-instance.

In the other case, we remove at most k disjoint obstructions. By Lemma 15, each obstruction contains at most $3d + 2$ vertices. Hence, the total number of edges added to S is at most $k(9d^2 + 12d + 4) = \mathcal{O}(d^2 k)$. This is indeed an $\mathcal{O}(d^2)$ approximate solution for Edge Deletion to d-VCC. Moreover, since $G - P$ contains no induced obstruction, we have $\mathtt{vc}_c(G - P) \leq d$, and therefore $G - P \in \mathcal{G}_d$.

Finally, since we invoke Theorem 3 at most $k + 1$ times, the overall running time is bounded accordingly. This concludes the proof. ◀

5.2 Marking and Irrelevant Vertices

From this point onward, we assume that we are given a set $P \subseteq E(G)$ of size $\mathcal{O}(d^2 k)$ such that $\mathtt{vc}_c(G - P) \leq d$. We now examine the connected components of the graph $G - P$. Let $\mathcal{CC}(G - P) = \{C_1, C_2, \ldots, C_\ell\}$ denote the set of connected components of $G - P$. Note that for each $C_i \in \mathcal{CC}(G - P)$, where $i \in [\ell]$, we have $\mathtt{vc}(C_i) \leq d$. For each $i \in [\ell]$, we compute a minimum vertex cover $Z_i \subseteq V(C_i)$ using the best-known FPT algorithm for Vertex Cover parameterized by the solution size d, which runs in time $\mathcal{O}(1.253^d \cdot d^{\mathcal{O}(1)} \cdot n)$ (see Proposition 17). We define the set

$$I_i \coloneqq V(C_i) \setminus (Z_i \cup V(P)),$$

Let k_i denote the number of edges in P that are incident to at least one vertex in C_i. Then $\sum_i k_i$ is at most $2|P|$, since each edge in P can be charged to at most two distinct components of $G - P$ in the summation, and no edge outside P contributes to $\sum_i k_i$. We now describe a marking scheme that will be used to identify irrelevant vertices, enabling their removal from G in order to obtain a kernel for the instance (G, k).

▶ **Marking Scheme 2.** *The marking scheme is defined as follows:*

(1) *For all $i \in [\ell]$, we do the following. For each pair $u, u' \in Z_i$ with $u \neq u'$, mark $k_i + 1$ common neighbors of u and u' in I_i.*

(2) *For all $i \in [\ell]$ and $v \in Z_i$, mark $k_i + d + 1$ neighbors of v in I_i.*

Let Mark be the set of marked vertices obtained from the above marking scheme.

▶ **Reduction Rule 5.** *Let $U = \bigcup_{i=1}^{t} I_i \setminus \mathsf{Mark}$ be the set of unmarked vertices in $\bigcup_{i=1}^{t} I_i$. The new instance is $(G - U, k)$.*

For the correctness of Reduction Rule 5, we state and prove the following lemmata.

▶ **Lemma 27 (★).** *Let (G, k) be a* Yes*-instance of* Edge Deletion to *d-VCC, and let $P \subseteq E(G)$ be a solution such that $\mathtt{vc}_c(G - P) \leq d$. Let $C_1, \ldots, C_\ell$ be the connected components of $G - P$. Then there exists a solution $S \subseteq E(G)$ of size at most k such that:*

$$|S \cap E(C_i)| \leq k_i \quad \text{for all } i \in [\ell],$$

where k_i denotes the number of edges in P that are incident to at least one vertex in $C_i \in \mathcal{CC}(G - P)$.

Finally, we present a lemma that establishes the safeness of the reduction rule.

▶ **Lemma 28 (★).** *(G, k) is a* Yes*-instance of* Edge Deletion to *d-VCC if and only if $(G - U, k)$ is a* Yes*-instance of* Edge Deletion to *d-VCC.*

Similar to implementation of Marking Scheme 1 and Reduction Rule 4, we can implement Marking Scheme 2 and Reduction Rule 5 in time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$.

▶ **Lemma 29.** *Reduction Rule 5 can be applied in time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$.*

▶ **Theorem 2.** *Edge Deletion to d-VCC admits a kernel with at most $\mathcal{O}(d^4k)$ vertices and $\mathcal{O}(d^5k)$ edges. Furthermore, it can be computed in time $\mathcal{O}\left(1.253^d \cdot (kd)^{\mathcal{O}(1)} \cdot n \log n\right)$.*

Proof. Let (G, k) be an instance of Edge Deletion to d-VCC. We begin by exhaustively removing all connected components C of G such that $\mathtt{vc}(C) \leq d$. These components already satisfy the target property and require no further edge deletions. After this preprocessing step, every remaining connected component must require at least one edge deletion. This can be performed in $\mathcal{O}(1.253^d \cdot d^{\mathcal{O}(1)} \cdot n)$ (see Proposition 17).

Next, we apply Reduction Rule 5 to further simplify the instance in time $\mathcal{O}((kd)^{\mathcal{O}(1)}n)$ by Lemma 29. We now analyze the structure of the remaining graph and proceed to bound its size.

- a d-sized vertex cover Z_i of C_i, for all $i \in [\ell]$
- the vertices of I_i those are marked by Marking Scheme 2, for all $i \in [\ell]$.
- $V(P)$.

Finally, we are ready to bound the number of vertices in the reduced graph. For a fixed $i \in [\ell]$, Marking Scheme 2 marks $\binom{|Z|}{2}(k_i+1) + |Z|(k_i+d+1)$ vertices, i.e., $\binom{d}{2}(k_i+1) + d(k_i+d+1)$ vertices of I_i. So, the total number of marked vertices in one connected component C_i of $G-P$ is at most $\binom{d}{2}(k_i+1)+d(k_i+d+1)$. Hence for each $i \in [\ell]$, there are $d+|V(P)\cap V(C_i)|+\binom{d}{2}(k_i+1)+d(k_i+d+1)$ vertices of C_i in the reduced graph. Summing over all the connected components C_i in $\mathcal{CC}(G-P)$, we get that the number of vertices in the reduced graph is

$$\sum_{i=1}^{\ell}\left(d+|V(P)\cap V(C_i)|+\binom{d}{2}(k_i+1)+d(k_i+d+1)\right) = \mathcal{O}(d^4k).$$

We now proceed to bound the number of edges. Let C_i be a connected component of $G-P$. There are at most $\mathcal{O}(d^2)$ edges in the vertex cover Z_i of C_i of size at most d. In C_i, $C_i - Z_i$ is an independent set and hence there are no edges in $C_i - Z_i$. Because of the reduction rule, the number of vertices in $C_i - Z_i$ is at most $d(k_i+d+1)+d^2(k_i+1)$, and therefore the number of edges between Z_i and $C_i - Z_i$ is at most $d(d(k_i+d+1)+d^2(k_i+1))$. So the total number of edges in component C_i is at most $d^2+d(d(k_i+d+1)+d^2(k_i+1))$. Hence, the total number of edges in the graph is $|P|+\sum_{i=1}^{|P|} d^2+d(d(k_i+d+1)+d^2(k_i+1))$, which is $\mathcal{O}(d^5k)$ (since $\sum_i k_i \leq 2|P|$). Hence, the number of edges is bounded by $\mathcal{O}(d^5k)$. This concludes the theorem. ◀

6 Conclusion

In this paper, we designed an almost-linear-time kernelization algorithm for clustering into components with bounded vertex cover number. An interesting direction for future work is to extend this framework to other notions of structural simplicity in clusters – such as bounded feedback vertex number or bounded odd cycle transversal.

Another natural question is whether our algorithm can be further optimized to achieve fully linear-time preprocessing. Additionally, designing a kernel for the vertex-deletion variant whose size is linear in the number of vertices remains an intriguing open problem.

References

1. Tara Abrishami, Maria Chudnovsky, Cemil Dibek, and Pawel Rzazewski. Polynomial-time algorithm for maximum independent set in bounded-degree graphs with no long induced claws. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1448–1470. SIAM, 2022. doi:10.1137/1.9781611977073.61.
2. Faisal N. Abu-Khzam. An improved kernelization algorithm for r-set packing. *Inf. Process. Lett.*, 110(16):621–624, 2010. doi:10.1016/J.IPL.2010.04.020.
3. Faisal N. Abu-Khzam. A kernelization algorithm for d-hitting set. *J. Comput. Syst. Sci.*, 76(7):524–531, 2010. doi:10.1016/J.JCSS.2009.09.002.
4. Noga Alon, Asaf Shapira, and Benny Sudakov. Additive approximation for edge-deletion problems. *Annals of Mathematics*, 170(1):371–411, 2009. URL: http://www.jstor.org/stable/40345467.
5. Matthias Bentert, Michael R. Fellows, Petr A. Golovach, Frances A. Rosamond, and Saket Saurabh. Breaking a graph into connected components with small dominating sets. In Rastislav Královic and Antonín Kucera, editors, *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024, August 26-30, 2024, Bratislava, Slovakia*, volume 306 of *LIPIcs*, pages 24:1–24:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.MFCS.2024.24.

6 Hans L. Bodlaender, Pinar Heggernes, and Daniel Lokshtanov. Graph modification problems (dagstuhl seminar 14071). *Dagstuhl Reports*, 4(2):38–59, 2014. doi:10.4230/DAGREP.4.2.38.

7 Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999. doi:10.1137/1.9780898719796.

8 Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.

9 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595.

10 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/S00453-015-0014-X.

11 Jianer Chen, Qin Huang, Iyad Kanj, and Ge Xia. Near-optimal algorithms for point-line covering problems. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 21:1–21:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.STACS.2022.21.

12 Jianer Chen, Qin Huang, Iyad Kanj, and Ge Xia. Nearly time-optimal kernelization algorithms for the line-cover problem with big data. *Algorithmica*, 86(8):2448–2478, 2024. doi:10.1007/S00453-024-01231-6.

13 Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001. doi:10.1006/JAGM.2001.1186.

14 Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $o(n^2)$ steps. In Juraj Hromkovic, Manfred Nagl, and Bernhard Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science, 30th International Workshop,WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers*, volume 3353 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2004. doi:10.1007/978-3-540-30559-0_22.

15 Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *Comput. Sci. Rev.*, 48:100556, 2023. doi:10.1016/J.COSREV.2023.100556.

16 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.

18 Michael R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In Hans L. Bodlaender, editor, *Graph-Theoretic Concepts in Computer Science, 29th International Workshop, WG 2003, Elspeet, The Netherlands, June 19-21, 2003, Revised Papers*, volume 2880 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003. doi:10.1007/978-3-540-39890-5_1.

19 F.V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

20 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

21 David G. Harris and N. S. Narayanaswamy. A faster algorithm for vertex cover parameterized by solution size. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France*, volume 289 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.STACS.2024.40.

22 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.

23 Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010. doi:10.1007/S00224-008-9150-X.

24 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.

25 Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. *Theory Comput. Syst.*, 53(2):263–299, 2013. doi:10.1007/S00224-012-9393-4.

26 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811. SIAM, 2014. doi:10.1137/1.9781611973402.130.

27 George Karypis, Christian Schulz, Darren Strash, Deepak Ajwani, Rob H. Bisseling, Katrin Casel, Ümit V. Çatalyürek, Cédric Chevalier, Florian Chudigiewitsch, Marcelo Fonseca Faraj, Michael R. Fellows, Lars Gottesbüren, Tobias Heuer, Kamer Kaya, Jakub Lacki, Johannes Langguth, Xiaoye Sherry Li, Ruben Mayer, Johannes Meintrup, Yosuke Mizutani, François Pellegrini, Fabrizio Petrini, Frances A. Rosamond, Ilya Safro, Sebastian Schlag, Roohani Sharma, Blair D. Sullivan, Bora Uçar, and Albert-Jan Yzelman. Recent trends in graph decomposition (dagstuhl seminar 23331). *Dagstuhl Reports*, 13(8):1–45, 2023. doi:10.4230/DAGREP.13.8.1.

28 Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 382–390. ACM, 2007. doi:10.1145/1250790.1250848.

29 Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discret. Appl. Math.*, 160(15):2259–2270, 2012. doi:10.1016/J.DAM.2012.05.019.

30 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

31 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

32 N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. LP can be a cure for parameterized problems. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 338–349. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPICS.STACS.2012.338.

33 George L. Nemhauser and Leslie E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974. doi:10.1007/BF01580222.

34 René van Bevern. Towards optimal and expressive kernelization for d-hitting set. *Algorithmica*, 70(1):129–147, 2014. doi:10.1007/S00453-013-9774-3.

35 Mihalis Yannakakis. Edge-deletion problems. *SIAM J. Comput.*, 10(2):297–309, 1981. doi:10.1137/0210021.