Feedback Vertex Set Inspired Kernel for Chordal Vertex Deletion*

Akanksha Agrawal† Daniel Lokshtanov‡ Pranabendu Misra§ Saket Saurabh¶
Meirav Zehavi‖

Abstract

Given a graph G and a parameter k, the Chordal Vertex Deletion (CVD) problem asks whether there exists a subset $U \subseteq V(G)$ of size at most k that hits all induced cycles of size at least 4. The existence of a polynomial kernel for CVD was a well-known open problem in the field of Parameterized Complexity. Recently, Jansen and Pilipczuk resolved this question affirmatively by designing a polynomial kernel for CVD of size $\mathcal{O}(k^{161}\log^{58} k)$, and asked whether one can design a kernel of size $\mathcal{O}(k^{10})$. While we do not completely resolve this question, we design a significantly smaller kernel of size $\mathcal{O}(k^{25}\log^{14} k)$, inspired by the $\mathcal{O}(k^2)$-size kernel for Feedback Vertex Set. To obtain this result, we first design an $\mathcal{O}(\mathsf{opt}\cdot\log^2 n)$-factor approximation algorithm for CVD, which is central to our kernelization procedure. Thus, we improve upon both the kernelization algorithm and the approximation algorithm of Jansen and Pilipczuk. Next, we introduce the notion of the independence degree of a vertex, which is our main conceptual contribution. We believe that this notion could be useful in designing kernels for other problems.

1 Introduction

Data reduction techniques are widely applied to deal with computationally hard problems in real world applications. It has been a long-standing challenge to formally express the efficiency and accuracy of these "preprocessing" procedures. The framework of parameterized complexity turns out to be particularly suitable for a mathematical analysis of pre-processing heuristics. Formally, in parameterized complexity each problem instance is accompanied by a parameter k, and we say that a parameterized problem is *fixed-parameter tractable (FPT)* if there is an algorithm that solves the problem in time $f(k)\cdot|I|^{\mathcal{O}(1)}$, where $|I|$ is the size of the input and f is a computable function of the parameter k alone. *Kernelization* is the subarea of parameterized complexity that deals with the mathematical analysis of pre-processing heuristics. A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial-time algorithm (the degree of polynomial is independent of the parameter k), called a *kernelization algorithm*, that reduces the input instance down to an instance whose size is bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$-*kernel* for the problem. Observe that if a problem has a kernelization algorithm, then it is also has an FPT algorithm.

Kernelization appears to be an interesting computational approach not only from a theoretical perspective, but also from a practical perspective. There are many real-world applications where even very simple preprocessing can be surprisingly effective, leading to significant size-reduction of the input. Kernelization is a natural tool for measuring the quality of existing preprocessing rules proposed for specific problems as well as for designing new powerful such rules. The most fundamental question in the field of kernelization is:

> Let Π be parameterized problem that admits an FPT algorithm. Then, does Π admit a polynomial kernel?

In recent times, the study of kernelization, centred on the above question, has been one of the main areas of research in parameterized complexity, yielding many new important contributions to theory. These include general results showing that certain classes of parameterized problems have polynomial kernels, and as well as other results that utilize advanced techniques from algebra, matroid theory and topology for data reduction [3, 5, 19, 18, 23, 26, 29, 30, 33, 32, 41, 42]. The development of a framework for ruling out

*The research leading to these results received funding from the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreements no. 306992.

†University of Bergen, Bergen, Norway. akanksha.agrawal@uib.no.

‡University of Bergen, Bergen, Norway. daniello@uib.no.

§Institute of Mathematical Sciences, HBNI, Chennai, India. pranabendu@imsc.res.in.

¶University of Bergen, Bergen, Norway, and the Institute of Mathematical Sciences, HBNI, Chennai, India. saket@imsc.res.in.

‖University of Bergen, Bergen, Norway. meirav.zehavi@uib.no.

polynomial kernels under certain complexity-theoretic assumptions [4, 6, 13, 21] has added a new dimension to the area, and strengthened its connections to classical complexity theory. We refer to the following surveys [31, 36] and the corresponding chapters in the books [11, 14, 17, 40], for a detailed introduction to the field of kernelization.

An important class of problems, that has led to the development of many upper bound tools and techniques in kernelization, is the class of *parameterized graph deletion problems*. A typical problem of this class is associated with a family of graphs, $\mathcal{F}$, such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs or planar graphs. The deletion problem corresponding to $\mathcal{F}$ is formally stated as follows.

$\mathcal{F}$-Vertex (Edge) Deletion **Parameter:** k **Input:** An undirected graph G and a non-negative integer k. **Question:** Does there exist $S \subseteq V(G)$ (or $S \subseteq E(G)$) such that $\lvert S\rvert \leq k$ and $G \setminus S$ is in $\mathcal{F}$?

Graph deletion problems are also among the most basic problems in graph theory and graph algorithms. Most of these problems are NP-complete [45, 35], and thus they were subject to intense study in various algorithmic paradigms to cope with their intractability [22, 37, 18, 39]. These include, considering a restricted class of inputs, approximation algorithms, parameterized complexity and kernelization.

Some of the most well known results in kernelization are polynomial kernels for graph deletion problems such as Feedback Vertex Set [42], Odd Cycle Transversal [33, 32], Vertex Cover [2, 10], Planar-$\mathcal{F}$-Deletion [18], and Treedepth-η-Deletion [23]. A common thread among all these problems, with the exception of Odd Cycle Transversal, is that the corresponding family $\mathcal{F}$ can be characterized by a finite set of forbidden minors that include at least one connected planar graph. It is known that, if $\mathcal{F}$ can be characterized by a finite set of forbidden induced subgraphs, then the corresponding $\mathcal{F}$-Vertex Deletion problem immediately admits an FPT algorithm as well as polynomial sized kernel because of its connection to d-Hitting Set [1]. However, if $\mathcal{F}$ is characterized by an *infinite* set of forbidden induced subgraphs, which is the case when $\mathcal{F}$ is the class of chordal graphs, chordal bipartite graphs, interval graphs, proper interval graphs and permutation graphs, our understanding of these problems in the realm of parameterized complexity and kernelization, is still at a nascent stage. While Chordal Vertex Deletion (CVD) was known to be FPT for some time [38, 9], the parameterized complexity of Interval Vertex Deletion was settled only recently [8, 7]. The parameterized complexity of Permutation Vertex Deletion and Chordal Bipartite Vertex Deletion is still unknown. Coming to the question of polynomial kernels for these problems, the situation is even more grim. Until recently, the only known result was a polynomial kernel for Proper Interval Vertex Deletion: Fomin et al. [20] obtained a $\mathcal{O}(k^{53})$ sized polynomial kernel for Proper Interval Vertex Deletion, which has recently been improved to $\mathcal{O}(k^4)$ [28]. A dearth of further results in this area has led to the questions of kernelization complexity of Chordal Vertex Deletion and Interval Vertex Deletion becoming prominent open problems [9, 12, 20, 25, 38].

Jansen and Pilipzuck [27] recently resolved one of these open questions. They showed that CVD admits a polynomial kernel of size $\mathcal{O}(k^{161}\log^{58} k)$, and further posed an open question: Does CVD admit a kernel of size $\mathcal{O}(k^{10})$? While we do not completely resolve this question, we design a significantly smaller kernel. In particular, we get the following result.

Theorem 1.1. CVD *admits a polynomial kernel of size* $\mathcal{O}(k^{25}\log^{14} k)$.

Our Methods. Our result is inspired by the $\mathcal{O}(k^2)$-size kernel for Feedback Vertex Set (FVS) (checking whether there exists a k sized vertex subset that intersects all cycles) , designed by Thomassé [42]. The kernel for FVS consists of the two following steps.

1. Reduce the maximum degree of the graph by using matching based tools (in particular expansion lemma). That is, upper bound the maximum degree Δ of the graph by $\mathcal{O}(k)$.
2. When the graph has maximum degree Δ, one can show that if a graph has minimum degree at least 3 (which can be easily achieved for FVS) then any minimum feedback vertex set has size $\mathcal{O}(n/\Delta)$. This together with an upper bound on Δ implies $\mathcal{O}(k^2)$-size kernel.

Let us now look at the CVD problem. Here, our objective is to check whether there exists a k-sized vertex subset S such that $G \setminus S$ is a chordal graph. However, what is a chordal graph? A graph is chordal if it does not contain any induced cycle of length at least 4. That is, every cycle of length at least 4 has a chord. Thus, FVS is about intersecting all cycles and CVD is about intersecting all chordless cycles. Unfortunately, this apparent similarity stops here! Nevertheless, we are still able to exploit ideas used in the $\mathcal{O}(k^2)$-size kernel for FVS. Towards this, we define the notion of *independence degree* of vertices and

graphs. Roughly speaking, the independence degree of a vertex v is the size of a maximum independent set in its neighborhood ($G[N(v)]$). The study of this notion is our main conceptual contribution.

As a first step we bound the independence degree of every vertex by $k^{\mathcal{O}(1)}$ – this is similar to the first step of the kernel for FVS. Once we have bounded the independence degree of a graph, we obtain an approximate solution M (also called modulator) and analyze the graph $G \setminus M$. The bound on the independence degree immediately implies that the number of leaves, vertices of degree at least three and the number of maximal degree two paths in the clique forest of $G \setminus M$ is bounded by $k^{\mathcal{O}(1)}$. Then, using ideas similar to those used by Marx [38] to bound the size of a maximal clique while designing the first algorithm for CVD, we reduce the size of each maximal clique in $G \setminus M$ to $k^{\mathcal{O}(1)}$. Finally, we use structural analysis to bound the size of each maximal degree two path, which includes the design of a reduction rule that computes a family of minimum cuts, and thus we obtain our final kernel. We believe that the notion of independent degree is likely to be useful for designing algorithms for other graph modification problems. Not only does it lead to a significant improvement, once it is bounded, it greatly simplifies the analysis of the resulting instance.

Approximation Algorithm. An important issue that we did not address in the previous paragraph concerns the manner in which we obtain the modulator M. Towards this, Jansen and Pilipczuk [27] designed a polynomial-time $\mathcal{O}(\mathsf{opt}^2 \log \mathsf{opt} \log n)$-factor approximation algorithm for CVD. Our second main contribution is an improved approximation algorithm for CVD. More precisely, we obtain the following theorem.

THEOREM 1.2. CVD *admits a polynomial-time* $\mathcal{O}(\mathsf{opt} \log^2 n)$*-factor approximation algorithm.*

Our approximation algorithm is based on the method of divide-and-conquer. On the one hand, the application of this method is guided by the structure of a "tree decomposition" satisfying an invariant that ensures that relations between unresolved subinstances are "simple". On the other hand, the analysis of this method is guided by a measure which captures, in an exploitable sense, the complexity of the instance at hand. More precisely, we identify "complex subinstances" and isolate them using a *small* number of cliques (two per complex subinstance), and, roughly speaking, we deduce that an instance is complex if it contains many vertices that belong to complex subinstances and these complex subinstances are not well divided. The complex instances are handled, in one case, using hitting set-based reduction rules, and in another case, using computations of balanced separators and multiway cuts.

Finally, after we obtain a kernel, we show that by reruning our entire kernelization procedure once, we can actually reduce the size of the kernel. When we call our kernelization procedure for the first time, we work with an approximate solution of size $\mathcal{O}(k^4 \log^2 k)$; indeed, it can be assumed that $\log n < k \log k$, else the $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$-time algorithm for CVD by Cao and Marx [9] solves the input instance in polynomial time. However, once we have our first kernel, it holds that $n = \mathcal{O}(k^{41} \log^{18} k)$. At this point, if we reuse our approximation algorithm, we obtain an approximate solution of size $\mathcal{O}(k^2 \log^2 k)$. The size of our kernel depends on the size of the approximate solution; in particular, having an approximate solution of size $\mathcal{O}(k^2 \log^2 k)$ allows us to obtain a kernel of size $\mathcal{O}(k^{25} \log^{14} k)$.

2 Preliminaries

For a positive integer k, we use $[k]$ as a shorthand for $\{1, 2, \ldots, k\}$.

Parameterized Complexity. In Parameterized Complexity each problem instance is accompanied by a parameter k. A central notion in this field is the one of *fixed-parameter tractability (FPT)*. This means, for a given instance (I, k), solvability in time $f(k)|I|^{\mathcal{O}(1)}$ where f is some computable function of k. A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial-time algorithm (the degree of polynomial is independent of the parameter k), called a *kernelization algorithm*, that reduces the input instance down to an equivalent instance whose size is bounded by a polynomial $p(k)$ in k. Here, two instances are equivalent if one of them is a yes-instance if and only if the other one is a yes-instance. The reduced instance is called a $p(k)$*-kernel* for the problem. For a detailed introduction to the field of kernelization, we refer to the following surveys [31, 36] and the corresponding chapters in the books [11, 14, 17, 40].

Kernelization algorithms often rely on the design of *reduction rules*. The rules are numbered, and each rule consists of a condition and an action. We always apply the first rule whose condition is true. Given a problem instance (I, k), the rule computes (in polynomial time) an instance (I', k') of the same problem where $k' \leq k$. Typically, $|I'| < |I|$, where if this is not the case, it should be argued why the rule can be applied only polynomially many times. We say that the rule *safe* if the instances (I, k) and (I', k') are equivalent.

Graphs. Given a graph G, we let $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. In

this paper, we only consider undirected graphs. We let $n = |V(G)|$ denote the number of vertices in the graph G, where G will be clear from context. The *open neighborhood*, or simply the *neighborhood*, of a vertex $v \in V(G)$ is defined as $N_G(v) = \{w \mid \{v, w\} \in E(G)\}$. The *closed neighborhood* of v is defined as $N_G[v] = N_G(v) \cup \{v\}$. The *degree* of v is defined as $d_G(v) = |N_G(v)|$. We can extend the definition of neighborhood of a vertex to a set of vertices as follows. Given a subset $U \subseteq V(G)$, $N_G(U) = \bigcup_{u \in U} N_G(u)$ and $N_G[U] = \bigcup_{u \in U} N_G[u]$. The *induced subgraph* $G[U]$ is the graph with vertex-set U and edge-set $\{\{u, u'\} \mid u, u' \in U, \text{ and } \{u, u'\} \in E(G)\}$. Moreover, we define $G \setminus U$ as the induced subgraph $G[V(G) \setminus U]$. We omit subscripts when the graph G is clear from context. An *independent set* in G is a set of vertices such that there is no edge between any pair of vertices in this set. The *independence number* of G, denoted by $\alpha(G)$, is defined as the cardinality of the largest independent set in G. A *clique* in G is a set of vertices such that there is an edge between every pair of vertices in this set.

A *path* P in G is a subgraph of G where $V(P) = \{x_1, x_2, \ldots, x_\ell\} \subseteq V(G)$ and $E(P) = \{\{x_1, x_2\}, \{x_2, x_3\}, \ldots, \{x_{\ell-1}, x_\ell\}\} \subseteq E(G)$ for some $\ell \in [n]$. The vertices x_1 and x_ℓ are called *endpoints* of the path P and the remaining vertices in $V(P)$ are called *internal vertices* of P. We also say that P is a path between x_1 and x_ℓ. A *cycle* C in G is a subgraph of G where $V(C) = \{x_1, x_2, \ldots, x_\ell\} \subseteq V(G)$ and $E(C) = \{\{x_1, x_2\}, \{x_2, x_3\}, \ldots, \{x_{\ell-1}, x_\ell\}, \{x_\ell, x_1\}\} \subseteq E(G)$, i.e., it is a path with an additional edge between x_1 and x_ℓ. Let P be a path in the graph G on at least three vertices. We say that $\{u, v\} \in E(G)$ is a *chord* of P if $u, v \in V(P)$ but $\{u, v\} \notin E(P)$. Similarly, for a cycle C on at least four vertices, $\{u, v\} \in E(G)$ is a chord of C if $u, v \in V(C)$ but $\{u, v\} \notin E(C)$. A path P or cycle C is *chordless* if it has no chords.

The graph G is *connected* if there is a path between every pair of vertices, otherwise G is *disconnected*. A connected graph without any cycles is a *tree*, and a collection of trees is a *forest*. A maximal connected subgraph of G is called a *connected component* of G. Given a weight function $w : V(G) \to \{0, 1\}$, we say that a subset $U \subseteq V(G)$ is a *balanced separator for G with respect to w* if for each connected component C in $G \setminus U$ it holds that $\sum_{v \in V(C)} w(v) \leq \frac{2}{3} \sum_{v \in V(G)} w(v)$.

Forest Decompositions. A *forest decomposition* of a graph G is a pair (F, β) where F is forest, and $\beta : V(T) \to 2^{V(G)}$ is a function that satisfies the following,

(i) $\bigcup_{v \in V(F)} \beta(v) = V(G)$,

(ii) for any edge $\{v, u\} \in E(G)$ there is a node $w \in V(F)$ such that $v, u \in \beta(w)$,

(iii) and for any $v \in V(G)$, the collection of nodes $T_v = \{u \in V(F) \mid v \in \beta(u)\}$ is a subtree of F.

For $v \in V(F)$, we call $\beta(v)$ the *bag* of v, and for the sake of clarity of presentation, we sometimes use v and $\beta(v)$ interchangeably. We refer to the vertices in $V(F)$ as nodes. A *tree decomposition* is a forest decomposition where F is a tree.

Chordal Graphs. A graph G is a *chordal graph* if it has no chordless cycle as an induced subgraph, i.e., every cycle of length at least four has a chord. A *clique forest* of G is a forest decomposition of G where every bag is a maximal clique. The following lemma shows that the class of chordal graphs is exactly the class of graphs which have a clique forest.

LEMMA 2.1. (THEOREM 4.8, [24]) *A graph G is a chordal graph if and only if G has a clique forest.*

Given a subset $U \subseteq V(G)$, we say that U *hits* a chordless cycle C in G if $U \cap V(C) \neq \emptyset$. Observe that if U *hits* every chordless cycle of G, then $G \setminus U$ is a chordal graph. Given a $v \in V(G)$, we say that a vertex-set $B \subseteq V(G) \setminus \{v\}$ is a *v-blocker* if B hits every chordless cycle in G. Observe that the set B must *not* contain the vertex v.

The Expansion Lemma. Let c be a positive integer. A *c-star* is a graph on $c + 1$ vertices where one vertex, called the center, has degree c, and all other vertices are adjacent to the center and have degree one. A *bipartite graph* is a graph whose vertex-set can be partitioned into two independent sets. Such a partition the vertex-set is called a *bipartition* of the graph. Let G be a bipartite graph with bipartition (A, B). A subset of edges $M \subseteq E(G)$ is called *c-expansion of A into B* if

(i) every vertex of A is incident to exactly c edges of M,

(ii) and M saturates exactly $c|A|$ vertices in B.

Note that a c-expansion saturates all vertices of A, and for each $u \in A$ the set of edges in M incident on u form a c-star. The following lemma allows us to compute a c-expansion in a bipartite graph. It captures a certain property of neighborhood sets which is very useful for designing kernelization algorithms.

LEMMA 2.2. ([42, 11]) *Let G be a bipartite graph with bipartition (A, B) such that there are no isolated vertices in B. Let c be a positive integer such that $|B| \geq c|A|$. Then, there are non-empty subsets $X \subseteq A$ and $Y \subseteq B$ such that*

- *there is a c-expansion from X into Y,*
- *and there is no vertex in Y that has a neighbor in $A \setminus X$, i.e. $N_G(Y) = X$.*

Further, the sets X and Y can be computed in polynomial time.

3 Approximation Algorithm

In this section we give our approximation algorithm. In particular, we prove Theorem 1.2. We will call our algorithm with a parameter k, starting with $k = 1$ and incrementing its value at each call, until our algorithm returns a solution of size $\mathcal{O}(k^2 \log^2 n)$. Thus, by developing an $\mathcal{O}(k \log^2 n)$-factor approximation algorithm, we obtain an $\mathcal{O}(\mathsf{opt} \log^2 n)$-factor approximation algorithm.

Our algorithm is based on the method of divide-and-conquer. On the one hand, the application of this method is guided by the structure of a forest decomposition satisfying an invariant that ensures that relations between unresolved subinstances are "simple". On the other hand, the analysis of this method is guided by a measure which captures, in an exploitable sense, the complexity of the instance at hand. First, in Section 3.1, we describe invariants maintained during the entire execution of our algorithm as well as the measure used to analyze its performance. In Section 3.2, we explain the initialization and termination phases. Next, in Section 3.3 we handle the case where the instance has already been divided into many subinstances that are "bad" and whose relations are "simple". Section 3.4 handles the more difficult case where the structure of the instance is not well understood. Here, to divide the instance in a manner that both decreases the measure and preserves the invariants we find *two* sets of vertices, one after another, which first divide the instance into relatively smaller subinstances and then makes sure that the relations between these subinstances are simple. Finally, in Section 3.5, we analyze the performance of our algorithm.

3.1 Invariants and Measure First of all, we will ensure that after the initialization phase, the graph G will never contain induced cycles on exactly four vertices. We call this invariant the *C_4-free invariant*. In particular, this guarantee ensures that the graph G will always contain only a small number of maximal cliques:

LEMMA 3.1. ([15, 43]) *The number of maximal cliques of a C_4-free graph G is bounded by $\mathcal{O}(n^2)$, and they can be enumerated in polynomial time using a polynomial delay algorithm.*

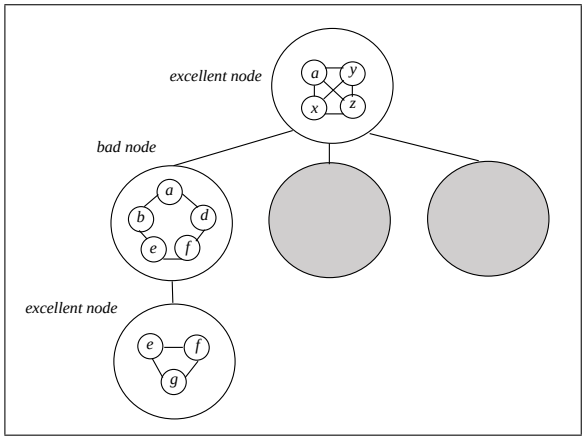

Figure 1: A bad node lies between two excellent nodes.

Moreover, throughout the execution of our algorithm, we will store a forest decomposition (F, β) of G. We categorize each node $v \in V(F)$ as follows: it is *bad* if $G[\beta(v)]$ is not a chordal graph; it is *good* if $G[\beta(v)]$ is a chordal graph that is not a clique; it is *excellent* if $G[\beta(v)]$ is a clique. With respect to (F, β), we maintain the following invariant: every node $v \in V(F)$ that is bad has at most two neighbors in F and both of these neighbors are excellent. We call this invariant the *division invariant*. An illustration of these notions is given in Fig. 1. We let V_b^F, V_g^F and V_e^F denote the sets of bad, good and excellent nodes, respectively. Furthermore, we let $V_{g,e}^G = \bigcup_{v \in V_g^F \cup V_e^F} \beta(v)$.

In fact, once we have a node that is good, we immediately turn it into an excellent node. To this end, we apply the following rule.

REDUCTION RULE 1. *If there exists a node $v \in V_g^F$, for each two non-adjacent vertices $u, w \in \beta(v)$, insert the edge $\{u, w\}$ into $E(G)$.*

In other words, Reduction Rule 1 turns subinstances that have already been resolved into cliques, which will simplify later arguments. It is clear that after its application, (F, β) remains a forest decomposition of G. Next, we argue that Reduction Rule 1 is safe in the sense that it preserves the invariants.

LEMMA 3.2. *Let G' be the graph resulting from the application of Reduction Rule 1. Then, both the C_4-free invariant and the division invariant are preserved. Moreover, a set S^* is an optimal solution to G if an only if it is an optimal solution to G'.*

To analyze the performance of our algorithm, we use the following measure: $\mu = \sum_{v \in V_b^F} |\beta(v) \setminus V_{g,e}^G|^2$. Here, the larger the value of the measure the more

difficult the instance seems to be. Intuitively, if it is complicated to solve the instance, the instance should have many vertices that belong to bad bags, while, at the same time, it should not have many bad bags (i.e., it should not be well divided). These two somewhat competing aspects of difficulty are both integrated into the definition of our measure. In particular, our measure is defined by a sum of squares of cardinalities rather than simply a sum of cardinalities to ensure that the measure decreases when we divide a bag asscoaited with a bad node into several smaller bags.

Observe that the application of Reduction Rule 1 does not change our measure. Thus, from now on, we can safely assume that V_g^F is empty.

3.2 Initialization and Termination To define (F, β), we simply let F be a tree on a single vertex to which β assigns $V(G)$. It is clear that the division invariant is satisfied. Observe that the measure, at this point, is n^2.

Now, we let S be an empty set. This set will contain the vertices we intend to insert into our approximate solution. As long as the graph G contains an induced cycle on exactly four vertices, we insert its four vertices into S. Since any optimal solution must hit every chordless cycle in G, we have the following observation.

Observation 1. *Any optimal solution S^* to G contains at least $\lceil |S|/4 \rceil$ vertices from S.*

Thus, we remove the vertices in S from G and decrease k by $\lceil |S|/4 \rceil$. Next, the C_4-free invariant is satisfied. It is also clear that the measure could not have increased.

We now turn to describe the case where we terminate the execution of our algorithm, which occurs when k drops to 0 or below or the value of the measure drops to 0. In case k drops to 0 or below, then if it drops exactly to 0 and $G \setminus S$ is a chordal graph, we return S as our approximate solution, and otherwise we conclude that there is no solution of size at most k. Otherwise, if the measure drops to 0, we have that (F, β) is a clique forest, and therefore the graph G is a chordal graph. Then, we return S as our approximate solution.

3.3 Many Bad Instances We now handle the simple case where $|V_b^F| \geq k+1$. Let $\widehat{V}_b^F$ denote an arbitrarily chosen subset of size exactly $k+1$ of V_b^F.

Observe that for each node $v \in \widehat{V}_b^F$, the subgraph $G[\beta(v)]$ contains a chordless cycle. We find one such chordless cycle, C_v, in polynomial time. We let $\mathcal{C} = \{C_v : v \in \widehat{V}_b^F\}$. It is clear that any chordless cycle can contain at most two vertices from any subgraph of G that is a clique. Thus, by the division invariant, we have the following simple yet very useful observation.

Observation 2. *Every cycle $C_v \in \mathcal{C}$ contains at most four vertices that do not belong* only *to the bag $\beta(v)$.*

For each cycle $C_v \in \mathcal{C}$, we let S_v denote the set of at most four vertices mentioned in Observation 2. We define $S' = \bigcup_{C_v \in \mathcal{C}} S_v$ and insert the vertices in S' into S. To justify this operation, we need the following lemma.

Lemma 3.3. *Any optimal solution S^* to G contains at least $\lceil |S'|/(4(k+1)) \rceil$ vertices from S'.*

Thus, we remove the vertices in S' from G and decrease k by $\lceil |S'|/(4(k+1)) \rceil$. Since we only removed vertices and did not update (F, β), it is clear that both of our invariants are still satisfied. Moreover, we could only have turned bad nodes into good or excellent nodes, and therefore the measure could not have increased.

3.4 Few Bad Instances We now handle the more difficult case where $|V_b^F| \leq k$. In this case there exists a node $v \in V_b^F$ such that $|\beta(v) \setminus V_{g,e}^G|^2 \geq \mu/k$. Clearly, we can find such a node v in polynomial time. Denote $n_v = |\beta(v) \setminus V_{g,e}^G|$ and $\mu_v = |\beta(v) \setminus V_{g,e}^G|^2$.

Balanced Vertex-Cut. We define a function $w : \beta(v) \to \{0, 1\}$ as follows. For each vertex $u \in \beta(v)$, let $w(u) = 1$ if $u \notin V_{g,e}^G$ and $w(u) = 0$ otherwise. Note that if there exists an optimal solution of size at most k, then there exists a set $S^* \subseteq \beta(v)$ of size at most k such that $G[\beta(v) \setminus S^*]$ is a chordal graph. Then, $G[\beta(v) \setminus S^*]$ admits a clique forest (F^*, β^*). In particular, every bag of F^* is a subset of the vertex-set of a maximal clique of $G[\beta(v)]$. Standard arguments on forests imply the correctness of the following observation.

Observation 3. *If there exists an optimal solution S^* of size at most k, then there exists a node $v^* \in V(F^*)$ such that $\beta^*(v^*)$ is a balanced separator for $G[\beta(v) \setminus S^*]$ with respect to w.*

Thus, we overall have the following observation.

Observation 4. *If there exists an optimal solution of size at most k, then there exist a maximal clique M of $G[\beta(v)]$ and a subset $M' \subseteq \beta(v) \setminus M$ of at most k vertices such that $M \cup M'$ is a balanced separator for $G[\beta(v)]$ with respect to w.*

The following lemma translates this observation into an algorithm.

LEMMA 3.4. *There is a polynomial-time algorithm such that if there exists an optimal solution of size at most k, it finds a maximal clique M of $G[\beta(v)]$ and a subset $M' \subseteq \beta(v) \setminus M$ of at most $\mathcal{O}(k \log n)$ vertices such that $M \cup M'$ is a balanced separator for $G[\beta(v)]$ with respect to w. In case it does not output such a pair (M, M'), there does not exist an optimal solution of size at most k.*

We remark that we used the $\mathcal{O}(\log n)$-factor approximation algorithm by Leighton and Rao [34] in Lemma 3.4 to find the balanced separator instead of the $\mathcal{O}(\sqrt{\log n})$-factor approximation algorithm by Feige et al. [16], as the algorithm by Feige et al. is randomized.

We call the algorithm in Lemma 3.4 to obtain a pair (M, M'). We insert the vertices in M' into S and remove M' from G. For simplicity of presentation, we abuse notation and refer to the graph resulting from the removal of M' by G. We remark that at this point, we have not yet updated (F, β) besides the removal of the vertices in M' from its bags. Now, observe that M is a balanced separator for $G[\beta(v)]$ with respect to w. Moreover, it is clear that both the C_4-free invariant and the division invariant are preserved. However, we cannot decrease k – we cannot argue that there exists an optimal solution that contains vertices from M'. Still, we will be able to "make progress" by updating our forest decomposition and consequently decreasing the measure. Roughly speaking, we cannot simply let M be a bag, and adjust the forest decomposition accordingly – in particular, the main obstacle in this operation is the fact that we need to preserve the division invariant while vertices (or sets of vertices that we would like to turn into bags) may be adjacent both to M and to vertices in both bags of the nodes that are the neighbors of v in F. To overcome this difficulty, we need to remove another set of vertices from the graph using the approach described below.

Preserving the Division Invariant: Simple Case. We let $\mathcal{A}$ denote the set of connected components of $G[\beta(v) \setminus M]$. Recall that we have shown that for each $A \in \mathcal{A}$ it holds that $|V(A) \setminus V^G_{g,e}| \leq 2n_v/3$. Observe that the node v has at most two neighbors. For the sake of simplicity of presentation, we assume w.l.o.g that v has exactly two neighbors, since if this is not the case, we can add "dummy" nodes to F to which β assigns the empty set and which are leaves adjacent only to v. We let u and w denote the two neighbors of v. Moreover, we denote $M_u = \beta(u) \cap \beta(v)$ and $M_w = \beta(w) \cap \beta(v)$. Recall that $G[M_u]$ and $G[M_w]$ are cliques. Thus, we have the following observation.

OBSERVATION 5. *There exists at most one connected component $A \in \mathcal{A}$ such that $V(A) \cap M_u \neq \emptyset$. Similarly, there exists at most one connected component $A \in \mathcal{A}$ such that $V(A) \cap M_w \neq \emptyset$.*

First, consider the simple case where there does not exist a connected component $A \in \mathcal{A}$ such that both $V(A) \cap M_u \neq \emptyset$ and $V(A) \cap M_w \neq \emptyset$. Then, we update the forest decomposition (F, β) of G to a forest decomposition $(\widehat{F}, \widehat{\beta})$ as follows. We start by removing v from F and adding three new vertices, $v_{u,M}$, v_M and $v_{M,w}$. We also add the edges $\{u, v_{u,M}\}, \{v_{u,M}, v_M\}, \{v_M, v_{M,w}\}, \{v_{M,w}, w\}$. We set $\widehat{\beta}(v_M) = M$. If there exists a connected component $A \in \mathcal{A}$ such that $V(A) \cap M_u \neq \emptyset$, then by Observation 5, there exists exactly one such component, which we denote by A_u. If such a component does not exist, we let A_u denote the empty graph. Symmetrically, we define A_w. Observe that A_u and A_w do not share any common vertex. We set $\widehat{\beta}(v_{u,M}) = V(A_u) \cup M$ and $\widehat{\beta}(v_{M,w}) = V(A_w) \cup M$. Clearly, v_M is an excellent node. Moreover, $v_{u,M}$ is adjacent only to two nodes, u and v_M, which are excellent nodes. Indeed, the node u is an excellent since it had been adjacent to v, which is a bad node, and before we updated the forest decomposition, the division invariant had been satisfied. Similarly, $v_{M,w}$ is adjacent only to two nodes, v_M and w, which are excellent nodes. For each connected component $A \in \mathcal{A} \setminus \{A_u, A_w\}$, we add a new node v_A and the edge $\{v_A, v_M\}$, and we set $\widehat{\beta}(v_A) = V(A) \cup M$. In this manner v_A is adjacent to exactly one node, which is an excellent node. Thus, by Observation 5, we have overall defined a valid forest decomposition which preserves the division invariant.

Observe that the vertex-set of each connected component in $\mathcal{A}$ is contained in a bag, and that each vertex v that belongs to some connected component in $\mathcal{A}$ and not to $M_u \cup M_w$ is contained in exactly one bag. Thus, the value of the measure μ has decreased by at least $\mu_v - \sum_{A \in \mathcal{A}} |V(A) \setminus (M_u \cup M_v)|^2 \geq n_v^2 - \max_{N \in \mathcal{N}} \sum_{n' \in N} n'^2$, where $\mathcal{N}$ is the family of all multisets N of positive integers such that $\sum_{n' \in N} n' = n_v$ and each $n' \in N$ satisfies $n' \leq 2n_v/3$. Observe that $n_v^2 - \max_{N \in \mathcal{N}} \sum_{n' \in N} n'^2 \geq n_v^2 - (2n_v/3)^2 - (n_v/3)^2 = 4n_v^2/9$. Thus, the measure decreased by at least $4n_v^2/9 = 4\mu_v/9$. Since we chose v such that $\mu_v \geq \mu/k$, we get that the measure decreased by at least $4\mu/(9k)$.

Preserving the Division Invariant: Difficult Case. Now, we consider the more difficult case where

there exists a connected component $A \in \mathcal{A}$ such that both $V(A) \cap M_u \neq \emptyset$ and $V(A) \cap M_w \neq \emptyset$. By Observation 5, there exists exactly one such component, which we denote by A', and for any other component $A \in \mathcal{A}$ it holds that $V(A) \cap (M_u \cup M_w) = \emptyset$. We denote $W = V(A') \cup M \cup M_u \cup M_w$.

Note that if there exists an optimal solution of size at most k, then there exists a set $S^* \subseteq \beta(v)$ of size at most k such that $G[W \setminus S^*]$ is a chordal graph admitting a clique forest (F^*, β^*). As before, every bag of F^* is a subset of the vertex-set of a maximal clique of $G[W]$. Observe that the clique forest F^* contains nodes a, b and c such that $M \setminus S^* \subseteq \beta^*(a), M_u \setminus S^* \subseteq \beta^*(b)$ and $M_w \setminus S^* \subseteq \beta^*(c)$, as well as a node d such that if we removed d from F^*, the nodes a, b and c would have belonged to different connected components (or removed from the graph in case one of them is d). In particular, by the definition of a clique forest, $\beta^*(d)$ is a maximal clique of $G[W \setminus S^*]$ such that the vertex-set of each connected component of $G[W \setminus (\beta^*(d) \cup S^*)]$ has nonempty intersection with at most one set in $\{M, M_u, M_w\}$. Thus, we have the following observation.

Observation 6. *If there exists an optimal solution of size at most k, then there exist a maximal clique $\widehat{M}$ of $G[W]$ and a subset $\widehat{M}' \subseteq W \setminus \widehat{M}$ of at most k vertices such that the vertex-set of every connected component C in $G[W \setminus \widehat{M} \cup \widehat{M}']$ has nonempty intersection with at most one set in $\{M, M_u, M_w\}$.*

The following lemma translates this observation into an algorithm.

Lemma 3.5. *There is a polynomial-time algorithm such that if there exists an optimal solution of size at most k, it finds a maximal clique $\widehat{M}$ of $G[W]$ and a subset $\widehat{M}' \subseteq W \setminus \widehat{M}$ of at most $4k/3$ vertices such that the vertex-set of every connected component C in $G[W \setminus \widehat{M} \cup \widehat{M}']$ has nonempty intersection with at most one set in $\{M, M_u, M_w\}$.*

We call the algorithm in Lemma 3.5 to obtain a pair $(\widehat{M}, \widehat{M}')$. We insert the vertices in $\widehat{M}'$ into S and remove $\widehat{M}'$ from G. Clearly, the C_4-free invariant is preserved. Next, let $\mathcal{B}$ denote the set of connected components of $G[W \setminus \widehat{M}]$. Then, by Lemma 3.5, the set $\mathcal{B}$ can be partitioned into three sets, $\mathcal{B}_M$, $\mathcal{B}_u$ and $\mathcal{B}_w$ such that each component in $\mathcal{B}_M$ does not contain vertices from $M_u \cup M_w$, each component in $\mathcal{B}_u$ does not contain vertices from $M \cup M_w$, and each component in $\mathcal{B}_w$ does not contain vertices from $M \cup M_u$. Observe that some of these sets may be empty.

Now, we update the forest decomposition (F, β) of G (which does not contain the vertices in $\widehat{M}'$) to a forest decomposition $(\widehat{F}, \widehat{\beta})$ as follows. We start by removing v from F and adding five new vertices, v_M, $v_{\widehat{M}}$, $v_{\widehat{M},M}$, $v_{\widehat{M},u}$ and $v_{\widehat{M},w}$. We also add the edges $\{u, v_{\widehat{M},u}\}, \{w, v_{\widehat{M},w}\}, \{v_M, v_{\widehat{M},M}\}, \{v_{\widehat{M}}, v_{\widehat{M},u}\}$, $\{v_{\widehat{M}}, v_{\widehat{M},w}\}$ and $\{v_{\widehat{M}}, v_{\widehat{M},M}\}$. We set $\widehat{\beta}(v_M) = M$, $\widehat{\beta}(v_{\widehat{M}}) = \widehat{M}$, $\widehat{\beta}(v_{u,\widehat{M}}) = (\bigcup_{B \in \mathcal{B}_u} V(B)) \cup \widehat{M}$, $\widehat{\beta}(v_{w,\widehat{M}}) = (\bigcup_{B \in \mathcal{B}_w} V(B)) \cup \widehat{M}$ and $\widehat{\beta}(v_{M,\widehat{M}}) = (\bigcup_{B \in \mathcal{B}_M} V(B)) \cup \widehat{M}$. Clearly, v_M and $v_{\widehat{M}}$ are excellent nodes. Furthermore, u and w were not changed, and therefore they remain excellent nodes. We thus have that each of the nodes $v_{\widehat{M},M}$, $v_{\widehat{M},u}$ and $v_{\widehat{M},w}$ is adjacent to exactly two nodes which are excellent nodes. For each connected component $A \in \mathcal{A} \setminus \{A'\}$, we add a new node v_A and the edge $\{v_A, v_M\}$, and we set $\widehat{\beta}(v_A) = V(A) \cup M$. In this manner v_A is adjacent to exactly one node, which is an excellent node. Thus, we have overall defined a valid forest decomposition which preserves the division invariant.

Finally, we remark that exactly in the same manner in which the decrease of the measure is analyzed in the previous case, we obtain that the measure decreases by at least $4\mu/(9k)$ in this case as well.

3.5 Approximation Ratio We are now ready to conclude that our algorithm is a polynomial-time $\mathcal{O}(k \log^2 n)$-factor approximation algorithm for CVD. It is clear that each divide-and-conquer iteration can be performed in polynomial time, and therefore to show that the algorithm runs in polynomial time, it is sufficient to bound to number of iterations it performs.

First, in the initialization phase, we insert some t vertices into S and decrease k by $\lceil t/4 \rceil$.[1] Now, observe that in Section 3.4 we decrease k by at least 1 and insert at most $4(k+1)$ vertices into S. In Section 3.4 we reduce the measure from μ to at most $(1 - 4/(9k))\mu$, and we insert at most $\mathcal{O}(k \log n)$ vertices into S. Note that initially μ is n^2 and the execution of the algorithm terminates once k drops to 0 or below or once μ drops to 0. Therefore, the algorithm can perform at most k iterations of the type described in Section 3.3, and by standard analysis (see, e.g, [44]), we get that it can perform at most $\mathcal{O}(k \log n)$ iterations of the type described in Section 3.4. Thus, we get that the algorithm performs $\mathcal{O}(k \log n)$ iterations, and at the end, if it returns an approximate solution, its size is bounded by $\mathcal{O}(k^2 \log^2 n)$. Recall that we have argued that if the algorithm does not return a solution, then there is no solution of size at most k. Thus, we conclude that our algorithm satisfies the desired properties.

[1] Recall that we have justified each decrease in the parameter k.

4 Kernelization

In this section we prove Theorem 1.1. First, we briefly state results relating to approximate solutions for CVD, which will be relevant later. Then, we address annotations that will be added to the input instance. Next, we introduce the notion of the *independent degree* of a vertex, which lies at the heart of the design of our kernelization algorithm. We carefully examine the independent degrees of vertices in our graphs, and show how these degrees can be bounded by a small polynomial in k. Afterwards, we consider the clique forest of the graph obtained by removing (from the input graph) the vertices of an approximate solution. In particular, we demonstrate the usefulness of our notion of an independent degree of a vertex – having bounded the independent degree of each vertex, we show that the number of leaves in the clique forest can be bounded in a simple and elegant manner. We also efficiently bound the size of a maximal clique. Then, we turn to bound the length of degree-2 paths in the clique forest. This part is quite technical, and its proofs are based on insights into the structure of chordal graphs and their clique forests. In particular, we use a reduction rule which computes a collection of minimum cuts rather than one minimum cut which overall allows us to capture the complexity of a degree-2 path using only few vertices. Finally, we use all these together with an alternating application of our approximation and kernelization algorithms to bound the size of our kernel.

Approximation. Observe that it can be assumed that $\log n < k \log k$, else the $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$-time algorithm for CVD by Cao and Marx [9] solves the input instance in polynomial time. Thus, based on Theorem 1.2, we obtain the following corollary.

Corollary 4.1. CVD *admits an* $\mathcal{O}(k^3 \log^2 k)$*-factor approximation algorithm.*

Throughout Section 4, we let APPROX denote a polynomial-time algorithm for CVD that returns approximate solutions of size $f(\mathsf{opt})$ for some function f. Initially, it will denote the algorithm given by Corollary 4.1. Given an instance of CVD, we say that an approximate solution D is *redundant* if for every vertex $v \in D$, $D \setminus \{v\}$ is also an approximate solution, that is, $G \setminus (D \setminus \{v\})$ is a chordal graph. Jansen and Pilipczuk [27] showed that given an approximate solution of size $g(k)$ for some function g, one can find (in polynomial time) either a vertex contained in every solution of size at most k or a redundant approximate solution of size $\mathcal{O}(k \cdot g(k))$. Thus, we have the following result.

Corollary 4.2. ([27]) *Given an instance of* CVD*, one can find (in polynomial time) either a vertex contained in every solution of size at most* k *or a redundant approximate solution of size* $\mathcal{O}(k \cdot f(k))$*.*

Next, we fix an instance (G, k) of CVD. By relying on APPROX and Corollary 4.2, we may assume that we have a vertex-set $\widetilde{D} \subseteq V(G)$ that is an approximate solution of size $f(k)$ and a vertex-set $D \subseteq V(G)$ that is a redundant approximate solution of size $c \cdot k \cdot f(k)$ for some constant c independent of the input. We also need to strengthen approximate solutions to be v-blockers for some vertices $v \in V(G)$. To this end, we will rely on the following result.

Lemma 4.1. *Given a vertex* $v \in V(G)$*, one can find (in polynomial time) either a vertex contained in every solution of size at most* k *or an approximate solution of size* $f(k)$ *that is a* v*-blocker.*

In light of Lemma 4.1, we may next assume that for every vertex $v \in V(G)$, we have a vertex-set B_v that is both a v-blocker and an approximate solution of size $f(k)$.

Irrelevant and Mandatory Edges. During the execution of our kernelization algorithm, we mark some edges in $E(G)$ as *irrelevant edges*. At the beginning of its execution, all of the edges in $E(G)$ are assumed to be relevant edges. When we mark an edge as an irrelevant edge, we prove that any solution that hits all of the chordless cycles in G that contain only relevant edges also hits all of the chordless cycles in G that contain the irrelevant edge. In other words, we prove that we can safely ignore chordless cycles that contain at least one irrelevant edge. Observe that we cannot simply remove irrelevant edges from $E(G)$ since this operation may introduce new chordless cycles in G. Instead we maintain a set E_I, which contains the edges marked as irrelevant. We also mark some edges in $E(G)$ as *mandatory edges*. We will ensure that at least one endpoint of a mandatory edge is present in any solution of size at most k. We let E_M denote the set of mandatory edges.

In some situations, we identify a pair of non-adjacent vertices such that any solution of size at most k must contain at least one of them. In this case we apply the following marking rule.

Reduction Rule 2. *Given two non-adjacent vertices in* G*,* v *and* u*, such that at least one of them belongs to any solution of size at most* k*, insert the edge* $\{v, u\}$ *into both* $E(G)$ *and* E_M*.*

Hence from now onwards our instance is of the form (G, k, E_I, E_M), and during the execution of our kernelization algorithm, we will update the sets E_I and

E_M. Later, we show that we can unmark the edges in $E_I \cup E_M$, obtaining an ordinary instance of CVD. For the sake of simplicity, when E_I and E_M are clear from context, we omit them.

If a vertex v is incident to at least $k+1$ mandatory edges, it must belong any solution of size at most k. Therefore, we may safely apply the following reduction rule.

Reduction Rule 3. *If there exists a vertex v incident to at least $k+1$ mandatory edges, remove v from G and decrement k by 1.*

After exhaustively applying the above reduction rule we have the following lemma.

Lemma 4.2. *If $|E_M| > k^2$ then the input instance is a no-instance.*

Thus, we will next assume that $|E_M| \leq k^2$ (else Reduction Rule 3 applies). Moreover, we let D' denote the set $D \cup \widetilde{D}$ to which we add every vertex that is an endpoint of a vertex in E_M (recall that $\widetilde{D}$ is our approximate solution of size at most $f(k)$ and D is our redundant approximate solution of size $\mathcal{O}(k \cdot f(k))$). Observe that by adding vertices to a redundant approximate solution, it remains a redundant approximate solution, and therefore D' or any other superset of D is such a solution.

Independent Degree. Given a vertex $v \in V(G)$, we use the notation $N^R_G(v)$ to refer to the set of each vertex $u \in N_G(v)$ such that $\{v, u\}$ does not belong to $E_I \cup E_M$. We start by introducing the notion of the independent degree of vertices and graphs.

Definition 1. *Given a vertex $v \in V(G)$, the* independent degree of v, *denoted by $d^I_G(v)$, is the size of a maximum independent set in the graph $G[N^R_G(v)]$. The* independent degree of G, *denoted by Δ^I_G, is the maximum independent degree of a vertex in $V(G)$.*

Fix $\Delta = (k+3)f(k)$. Next we investigate the notion of an independent degree, ultimately proving the following result.

Lemma 4.3. *One can construct (in polynomial time) an instance (G', k', E'_I, E'_R) of* CVD *that is equivalent to the input instance (G, k, E_I, E_R) and such that both $k' \leq k$ and $\Delta^I_{G'} \leq \Delta$.*

To this end, we assume that we are given a vertex $v \in V(G)$ such that $d^I_G(v) > \Delta$. We say that an instance (G', k', E'_I, E'_M) of CVD is *better* than the instance (G, k, E_I, E_M) if $k' \leq k$, $V(G') = V(G)$, $E_I \subseteq E'_I$, $E_M \subseteq E'_M$ $d^I_{G'}(v) \leq \Delta$ and for all $u \in V(G')$, $d^I_{G'}(u) \leq d^I_G(u)$. To prove the correctness of Lemma 4.3, it is sufficient to prove the correctness of the following lemma.

Lemma 4.4. *We can construct (in polynomial time) an instance (G', k', E'_I, E'_M) of* CVD *that is better than the input instance (G, k, E_I, E_M).*

Indeed, to prove Lemma 4.3, one can repeatedly apply the operation given by Lemma 4.4 in the context of every vertex $u \in V(G)$ such that $d^I_G(u) > \Delta$. We start with a simple result.

Lemma 4.5. *Let $u \in V(G)$ be a vertex such that $d^I_G(u) \geq |B_u|$. Then, one can find (in polynomial time) an independent set in $G[N^R_G(u) \setminus B_u]$ of size at least $d^I_G(u) - |B_u|$.*

Recall that for any vertex $u \in V(G)$, $|B_u| \leq f(k)$. Thus, we have the following corollary.

Corollary 4.3. *In polynomial time one can find an independent set in $G[N^R_G(v) \setminus B_v]$ of size at least $\Delta - f(k)$.*

We let I denote the independent set given by Corollary 4.3.

Independent Components. Let $X = N_G(v) \setminus (B_v \cup I)$ denote the neighbor-set of v from which we remove the vertices of the v-blocker B_v and of the independent set I. We also let $H = G \setminus (\{v\} \cup B_v \cup X)$ denote the graph obtained by removing (from G) the vertex v, the v-blocker B_v and any neighbor of v that does not belong to the independent set I. We define the set of independent components of v as the set of each connected component of H that contains at least one vertex from I, and denote this set by $\mathcal{A}$. For $\mathcal{A}$, we prove the following lemmas.

Lemma 4.6. *Each connected component $A \in \mathcal{A}$ contains exactly one vertex from I and no other vertex from $N_G(v)$.*

By Lemma 4.6, for each connected component $A \in \mathcal{A}$ we can let $z(A) \in I$ denote the unique neighbor of v in A. In fact, Corollary 4.3 and Lemma 4.6 imply that $\Delta - f(k) \leq |\mathcal{A}|$.

Lemma 4.7. *Every vertex $x \in X$ that is adjacent (in G) to some vertex $y \in V(A)$, where $A \in \mathcal{A}$, is also adjacent (in G) to the vertex $z(A)$.*

An illustration of the relations between v, B_v X and $\mathcal{A}$ is given in Fig. 2.

The Bipartite Graph $\widehat{H}$. To decrease $d^I_G(u)$, we will consider the bipartite graph $\widehat{H}$, which is defined as

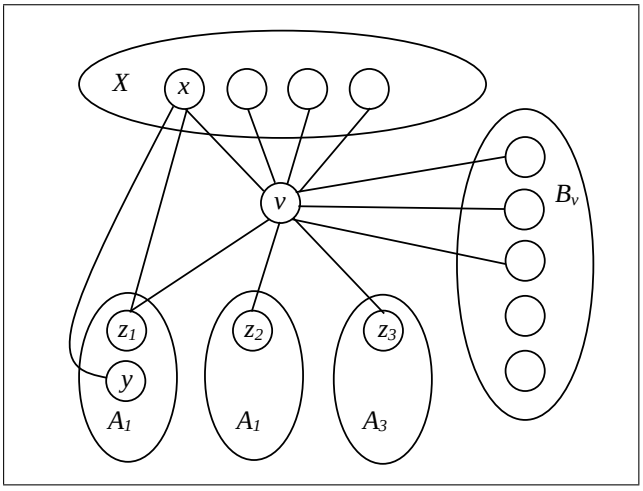

Figure 2: The relations between v, B_v, X and $\mathcal{A}$.

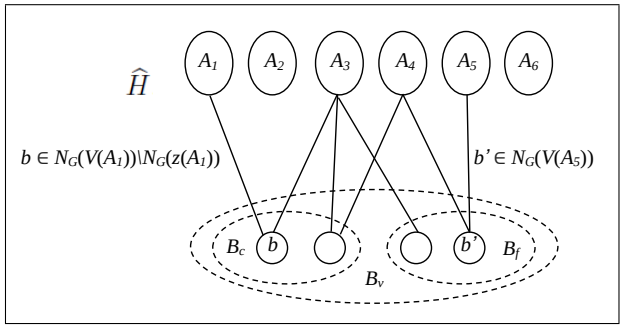

Figure 3: The bipartite graph $\widehat{H}$.

follows. We define vertex-set of $\widehat{H}$ by $V(\widehat{H}) = \mathcal{A} \cup B_v$. In this context, we mean that each connected component $A \in \mathcal{A}$ is represented by a vertex in $V(\widehat{H})$, and for the sake of simplicity, we use the symbol A also to refer to this vertex. We partition B_v into two sets, B_c and B_f, where B_c contains the vertices in B_v that are adjacent (in G) to v, while B_f contains the remaining vertices in B_v. Here, the letters c and f stand for "close" and "far". Having this partition, we define the edge-set of $\widehat{H}$ as follows. For every vertex $b \in B_c$ and connected component $A \in \mathcal{A}$ such that $b \in N_G(V(A)) \setminus N_G(z(A))$ (i.e., b is a neighbor of some vertex in A but not of the vertex $z(A)$), insert the edge $\{b, A\}$ into $E(\widehat{H})$. Moreover, for every vertex $b \in B_f$ and connected component $A \in \mathcal{A}$ such that $b \in N_G(V(A))$, insert the edge $\{b, A\}$ into $E(\widehat{H})$. An illustration of the bipartite graph $\widehat{H}$ is given in Fig. 3. The motivation behind its definition lies in the following lemma.

Lemma 4.8. *The bipartite graph $\widehat{H}$ satisfies the following properties.*

1. *Suppose that we are given an edge $\{b, A\} \in E(\widehat{H})$ such that $b \in B_c$ and $A \in \mathcal{A}$. Then, the graph G has a chordless cycle defined by the edges $\{v, b\}$ and $\{v, z(A)\}$ and a path in A.*
2. *Suppose that we are given edges $\{b, A\}, \{b, A'\} \in E(\widehat{H})$ such that $b \in B_f$ and $A, A' \in \mathcal{A}$. Then, the graph G has a chordless cycle defined by the edges $\{v, z(A)\}, \{v, z(A')\}$, the vertex b and paths in A and A'.*

Isolated Vertices in $\widehat{H}$. We start investigating the bipartite graph $\widehat{H}$ by examining the isolated vertices in $\mathcal{A}$ that it contains. In this context, we need the following lemma.

Lemma 4.9. *Let $A \in \mathcal{A}$ be an isolated vertex in $\widehat{H}$, and denote $z = z(A)$. Then, $N_G(V(A)) = N_G(z) \setminus V(A)$ and G does not have a chordless cycle that contains the edge $\{v, z\}$.*

Lemma 4.9 leads us to the design of the following reduction rule.

Reduction Rule 4. *If the graph $\widehat{H}$ contains an isolated vertex $A \in \mathcal{A}$, mark $\{v, z\}$ as irrelevant.*

After an exhaustive application of this rule, we can assume that $\widehat{H}$ does not contain an isolated vertex $A \in \mathcal{A}$.

Applying the Expansion Lemma. Next, we would like to apply Lemma 2.2 in the context of the bipartite graph $\widehat{H}$. Since $|\mathcal{A}| \geq \Delta - f(k) \geq (k+2) \cdot |B_v|$ and we have already ensured that $\widehat{H}$ does not contain any isolated vertex $A \in \mathcal{A}$, this lemma implies that we can find (in polynomial time) subsets $\mathcal{A}^* \subseteq \mathcal{A}$ and $B^* \subseteq B_v$ such that there exists a $(k+2)$-expansion from B^* into $\mathcal{A}^*$. The usefulness of B^* is stated in the following lemma.

Lemma 4.10. *Any solution of size at most k to the input instance that does not contain v contains all of the vertices in B^*.*

Decreasing the Independent Degree of v. Armed with Lemma 4.10, we can apply the following rule.

Reduction Rule 5. *For each vertex $b \in B^*$, insert the edge $\{b, v\}$ into $E(G)$ (if it is not already present), and mark $\{b, v\}$ as a mandatory edge. Moreover, mark each edge $\{v, z(A)\}$ such that $A \in \mathcal{A}^*$ as an irrelevant edge.*

Reduction Rule 5 decreases $|N_G^R(v)|$. Moreover, as long as $d_G^R(v) > \Delta$, we can apply this rule. Thus, after

an exhaustive application of this rule, it should hold that $d_G^R(v) \leq \Delta$. Furthermore, this rule neither inserts vertices into $V(G)$ nor unmarks edges, and therefore we conclude that Lemma 4.4 is correct. Denote $\Delta' = \Delta + 2k^2$. Thus, since we argued that we can assume that $|E_M| \leq k^2$, the size of a maximum independent set in the neighborhood of each vertex in the graph G from which we remove irrelevant edges is bounded by Δ'.

The Clique Forest. Let F denote the clique forest associated with the chordal graphs $G \setminus D'$. Towards bounding the number of leaves in F, we need the following lemma and the reduction rule.

LEMMA 4.11. *Let I be an independent set in the graph $G \setminus D'$. Then, there are most $|D'| \cdot \Delta'$ relevant edges between vertices in D' and vertices in I.*

REDUCTION RULE 6. *If there exists a vertex v in $G \setminus D'$ such that the vertices in $N_G(v)$ which are connected to v via relevant edges form a clique, remove the vertex v from G.*

The bound on the number of leaves will follow from a bound on the number of bags containing *private vertices*, which are defined as follows.

DEFINITION 2. *A vertex v in $G \setminus D'$ is a* private vertex *if there exists only one bag in F that contains it.*

LEMMA 4.12. *The number of bags in F containing private vertices is bounded by $|D'| \cdot \Delta'$.*

We now bound the number of leaves and nodes of degree at least 3 in the clique forest F.

LEMMA 4.13. *Both the number of leaves in F and the number of nodes of degree at least 3 in F are bounded by $|D'| \cdot \Delta'$.*

Next, we bound the size of a bag of F, to which end we prove the correctness of the following lemma.

LEMMA 4.14. *In polynomial time we can produce an instance (G', k') equivalent to (G, k) such that $k' \leq k$ and the size of any maximal clique in G' is bounded by $\kappa = c \cdot (|\widetilde{D}|^3 \cdot k + |\widetilde{D}| \cdot \Delta' \cdot (k+2)^3)$. (Here c is some constant independent of the input.)*

Observe that the size of each bag of F is bounded by the size of a maximal clique of $G \setminus D'$. Furthermore, since $G \setminus D'$ is a subgraph of G, the size of a maximal clique of $G \setminus D'$ is bounded by the size of a maximal clique of G. Using Lemma 4.14, we have the following result.

LEMMA 4.15. *The size of any bag of F is upper bounded by κ.*

The Length of Degree-2 Paths. Let V_F denote the set of each node of degree at least 3 in the forest F as well as each node whose bag has at least one private vertex. Let $\mathcal{P}$ denote the set of paths whose endpoints belong to V_F and such that all of their internal nodes do not belong to V_F. Clearly, it holds that $|\mathcal{P}| \leq |V_F|$. By Lemmas 4.12 and 4.13, we have the following observation: $|\mathcal{P}| \leq 2|D'| \cdot \Delta'$. Thus, in light of Lemma 4.15, by bounding the maximum number of nodes on each path in $\mathcal{P}$, we can bound the total number of vertices in the graph. To this end, we fix some path $P \in \mathcal{P}$. Moreover, we orient the path from left to right, where the choice of the leftmost and rightmost nodes is arbitrary.

Partitioning the Path P. Next, we will partition P into more "manageable paths". To this end, we need the following definition.

DEFINITION 3. *We say that a subpath Q of P complies with a vertex $d \in D'$ if one of the following conditions holds.*

- *For every two bags B and B' on Q, both $B \subseteq N_G(d)$ and $B' \subseteq N_G(d)$.*
- *For every two bags B and B' on Q, $B \cap N_G(d) = B' \cap N_G(d)$.*

We would like to find a set $\mathcal{B}$ of at most $\mathcal{O}(|D'| \cdot \kappa)$ bags on the path P such that after their removal from P, the following lemma will be true.

LEMMA 4.16. *Each subpath obtained by removing the bags in $\mathcal{B}$ from P complies with every vertex in D'.*

To prove Lemma 4.16, it is sufficient to show that for each vertex $d \in D'$, we can find $\mathcal{O}(\kappa)$ bags such that after their removal from P, each of the resulting subpaths complies with d. To this end, fix some vertex $d \in D'$.

LEMMA 4.17. *Let $u, v \in N_G(d) \setminus D'$ be non-adjacent vertices, B_u be a bag containing u such that no bag to its right (on P) contains u, and B_v be a bag containing v such that not bag to its left contains v. Then, d is adjacent to any vertex in any bag that lies strictly between B_u and B_v.*

We also need the following notation. Let B_ℓ be the leftmost bag on P that contains a neighbor v_ℓ of d such that v_ℓ does not belong to any bag to the right of B_ℓ. Similarly, let B_r be the rightmost bag on P that contains a neighbor v_r of d such that v_r does not belong to any bag to the left of B_ℓ.

Lemma 4.18. *Let B and B' be two bags on P to the left of B_ℓ such that B lies to the left of B'. Then, it holds that $B \cap N_G(d) \subseteq B' \cap N_G(d) \subseteq B_\ell \cap N_G(d)$.*

Lemma 4.19. *Let B and B' be two bags on P to the right of B_r such that B lies to the right of B'. Then, it holds that $B \cap N_G(d) \subseteq B' \cap N_G(d) \subseteq B_r \cap N_G(d)$.*

Let $B_1, B_2, \ldots, B_t$ be the set of bags that lie to the left of B_ℓ such that $B_1 \cap N_G(d) \subset B_2 \cap N_G(d) \subset \ldots \subset B_t \cap N_G(d) \subset B_\ell \cap N_G(d)$. By Lemma 4.18, this choice is well-defined, and it holds that each of the subpaths resulting from the removal of these bags from P, excluding the subpath that lies to the right of B_ℓ, complies with d. Moreover, by Lemma 4.15, $t = \mathcal{O}(\kappa)$. Symmetrically, by relying on Lemma 4.19, we handle the subpath of P that lies to the right of B_r. It remains to show that the subpath that lies strictly between B_ℓ and B_r (if this subpath exists) complies with d. However, the correctness of this claim is guaranteed by Lemma 4.17. Thus, we conclude that Lemma 4.16 is correct.

Handling a Manageable Path. We now examine a subpath of P, denoted by Q, which complies with every vertex $d \in D'$. We devise reduction rules such that after applying them exhaustively, the number of vertices in the union of the bags of the path Q is bounded by $\mathcal{O}(\kappa)$.

Let $B_1, B_2, \ldots, B_t$ denote the bags of Q ordered from left to right. Moreover, denote $V(Q) = \bigcup_{i=1}^{t} B_i$ and $A = \bigcap_{i=1}^{t} B_i$. We partition D' into two sets D_a and D_p, where $D_a = \{d \in D' : V(Q) \subseteq N_G(d)\}$ and $D_p = D' \setminus D_a$. Here the letters a and p stand for "all" and "partial" respectively. Since Q complies with every vertex $d \in D'$, for every vertex $d \in D_p$ and bags B_i, B_j, $i, j \in [t]$, it holds that $N_G(d) \cap B_i = N_G(d) \cap B_j$. In particular, this implies the following observation: $\bigcup_{d \in D_p} N_G(d) \subseteq A$. We denote $U = V(Q) \setminus (B_1 \cup B_t)$. It is sufficient to ensure that $|U| = \mathcal{O}(\kappa)$ since then, by Lemma 4.15, $|V(Q)| = \mathcal{O}(\kappa)$. Thus, we can next suppose that $|U| > \delta$ where $\delta = 2(k+1) + 6\kappa$. For each pair of non-adjacent vertices in D_a, we apply Reduction Rule 2. This is justified by the following lemma.

Lemma 4.20. *Let u and v be two distinct non-adjacent vertices in D_a. Then, every solution of size at most k contains at least one of the vertices u and v.*

Thus, from now on we can assume that $G[D_a]$ is a clique. However, by the definition of A, for every vertex in A and every vertex in $V(Q)$, there exists a bag B_i, $i \in [t]$, which contains both of them, and therefore they are adjacent. We thus deduce the following observation.

Observation 7. *Any two distinct vertices $v \in D_a \cup A$ and $u \in D_a \cup V(Q)$ are adjacent.*

Let us now examine chordless cycles that contain vertices from U.

Lemma 4.21. *Let C be a chordless cycle in G that contains some vertex $u \in U$. Then, no vertex on $V(C)$ belongs to $D_a \cup A$, and both neighbors of u in C do not belong to $D \cup A$.*

Lemma 4.22. *Let C be a chordless cycle in G that contains some vertex $u \in U$. Then, C contains a path between a vertex in $(B_1 \cap B_2) \setminus A$ and a vertex in $(B_{t-1} \cap B_t) \setminus A$ whose internal vertices belong to U and one of them is u.*

We continue to examine chordless cycles that contain vertices from U in the context of separators.

Lemma 4.23. *Let S be a minimal solution that contains at least one vertex from U. Then, there exists $i \in [t-1]$ such that $(B_i \cap B_{i+1}) \setminus A \subseteq S$ and $S \cap U \subseteq B_i \cap B_{i+1}$.*

Let $\mathcal{W}$ be the family of each subset $W \subseteq (B_1 \cup B_t) \setminus A$ of size at most k for which there exists an index $i \in [t-1]$ such that $W = (B_i \cap B_{i+1}) \setminus (A \cup U)$ and $|(B_i \cap B_{i+1}) \cap U| \leq k$. We can easily bound the size of the family $\mathcal{W}$ by $2k+1$. We proceed by associating a separator with each set $W \in \mathcal{W}$ as follows. First, let $\mathcal{I}_W$ denote the set of all indices $i \in [t-1]$ such that $W = (B_i \cap B_{i+1}) \setminus (A \cup U)$. Now, let i_W denote an index in $\mathcal{I}_W$ that minimizes $|(B_i \cap B_{i+1}) \cap U|$ (if there are several choices, choose one arbitrarily). We further denote $M = \bigcup_{W \in \mathcal{W}} ((B_{i_W} \cap B_{i_W+1}) \cap U)$. Observe that by Lemma 4.15 and the bound on $|\mathcal{W}|$ by $2k+1$, $|M| = \mathcal{O}(k^2)$. Thus, it is sufficient to argue that there exists a vertex in $U \setminus M$ that can be removed from G (since as long as $|U| > \delta$, we will be able to find such a vertex). To this end, we need the following.

Lemma 4.24. *Let $u \in U \setminus M$. If (G, k) is a yes-instance, then it has a solution S of size at most k that does not contain the vertex u.*

We now present the reduction rule that removes the irrelevant vertex found by Lemma 4.24.

Reduction Rule 7. *Let $u \in U \setminus M$. Remove the vertex u from the graph G and add an edge between any two non-adjacent vertices in $N_G(u)$.*

Finally, we unmark irrelevant edges and then apply the following rule to unmark mandatory edges.

Reduction Rule 8. *For every mandatory edge $\{x, y\}$ introduce $k+1$ pairs of new vertices, $\{x_1, y_1\}$, $\{x_2, y_2\}, \ldots, \{x_{k+1}, y_{k+1}\}$, and for each pair $\{x_i, y_i\}$ add the edges $\{x, x_i\}, \{x_i, y_i\}$ and $\{y_i, y\}$. Moreover, unmark the edge $\{x, y\}$.*

Since $|E_M| \leq k^2$, the total number of newly added vertices does not exceed $\mathcal{O}(k^3)$.

The Number of Vertices in the Kernel. In this section, we obtained an approximate solution $\widetilde{D}$ of size $f(k)$ and a redundant approximate solution D of size $\mathcal{O}(k \cdot f(k))$. Then, we examined the clique forest F associated with the chordal graph $G \setminus D'$ where $|D'| = \mathcal{O}(|D| + k^2)$. To this end, we considered a set $\mathcal{P}$ of degree-2 paths that together cover all of the nodes of the forest, and showed that $|\mathcal{P}| = \mathcal{O}(|D'| \cdot \Delta')$. Recall that $\Delta' = \mathcal{O}(k \cdot f(k))$. We removed $\mathcal{O}(|D'| \cdot \kappa)$ bags, each of size $\kappa = \mathcal{O}(|\widetilde{D}|^3 \cdot k + |\widetilde{D}| \cdot \Delta' \cdot k^3)$, from each path $P \in \mathcal{P}$, and considered each of the resulting subpaths Q. We showed the number of vertices in the union of the bags of the path Q will be bounded by $\mathcal{O}(\kappa)$. Finally, we added $\mathcal{O}(k^3)$ new vertices to unmark mandatory edges. Thus, we conclude that the number of vertices in our kernel is bounded by

$$\begin{aligned} & \mathcal{O}(|\mathcal{P}| \cdot |D'| \kappa \cdot \kappa) \\ = \; & \mathcal{O}(|D'|^2 \cdot \Delta' \cdot (|\widetilde{D}|^3 \cdot k + |\widetilde{D}| \cdot \Delta' \cdot k^3)^2) \\ = \; & \mathcal{O}((f(k)k)^3 \cdot (f(k)^3 k + f(k)^2 k^4)^2) \\ = \; & \mathcal{O}(f(k)^7 k^5 \cdot (f(k)^2 + k^6)) \end{aligned}$$

Recall that by Corollary 4.1, we can assume that $f(k) = \mathcal{O}(k^4 \log^2 k)$. Thus, at this point, we obtain a kernel of size $\mathcal{O}(k^{41} \log^{18} k)$.

A Better Kernelization Algorithm. Finally, we present a bootstrapping trick that will exploit the nature of our approximation algorithm to obtain a kernel of size $\mathcal{O}(k^{25} \log^{14} k)$. Recall that at this point, where we have already run our kernelization algorithm once, it holds that $n = \mathcal{O}(k^{41} \log^{18} k)$. Now, we again call our $\mathcal{O}(\mathsf{opt} \log^2 n)$-factor approximation algorithm (Theorem 1.2). Currently, it holds that $f(k) = \mathcal{O}(k^2 \log^2 k)$ rather than $f(k) = \mathcal{O}(k^4 \log^2 k)$. Thus, if we rerun our kernelization procedure, obtaining a kernel of size $\mathcal{O}(f(k)^7 k^5 \cdot (f(k)^2 + k^6))$ (see Section 4), it now holds that this size is upper bounded by $\mathcal{O}(k^{25} \log^{14} k)$. This concludes the proof of correctness of Theorem 1.1.

5 Conclusion

In this paper we obtained a polynomial kernel for CVD of size $\mathcal{O}(k^{25} \log^{14} k)$ and designed a factor $\mathcal{O}(\mathsf{opt} \log^2 n)$ approximation algorithm for CVD. The new kernel significantly improves over the previously known $\mathcal{O}(k^{161} \log^{58} k)$ sized kernel. On the other hand the approximation algorithm improves over the best previously known approximation algorithm for CVD, which had performance guarantee of $\mathcal{O}(\mathsf{opt}^2 \log \mathsf{opt} \log n)$. We believe that the notion of independence degree and the bootstrapping trick used in our kernelization procedure could be useful in designing polynomial kernels for other $\mathcal{F}$-Vertex (Edge) Deletion problems, where $\mathcal{F}$ is characterized by an infinite set of forbidden induced graphs. We conclude the paper with the following open problems.

- Design an approximation algorithm for CVD with factor $\mathcal{O}(\log^c n)$, for some fixed constant c. This will immediately reduce the size of our kernel to roughly $\mathcal{O}(k^{18})$.
- Does there exist an FPT algorithm for CVD with running time $c^k n^{\mathcal{O}(1)}$, for some fixed constant c?

References

[1] F. N. Abu-Khzam, *A kernelization algorithm for d-Hitting Set*, J. Comput. Syst. Sci., 76 (2010), pp. 524–531. 2

[2] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons, *Kernelization algorithms for the vertex cover problem: Theory and experiments*, in Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALC), SIAM, 2004, pp. 62–69. 2

[3] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *Solving MAX-r-SAT above a tight lower bound*, Algorithmica, 61 (2011), pp. 638–655. 1

[4] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin, *On problems without polynomial kernels*, J. Comput. Syst. Sci., 75 (2009), pp. 423–434. 2

[5] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos, *(meta) kernelization*, in Proc. 50th FOCS, IEEE Computer Society, 2009, pp. 629–638. 1

[6] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch, *Preprocessing for treewidth: A combinatorial analysis through kernelization*, SIAM J. Discrete Math., 27 (2013), pp. 2108–2142. 2

[7] Y. Cao, *Linear recognition of almost interval graphs*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, 2016, pp. 1096–1115. 2

[8] Y. Cao and D. Marx, *Interval deletion is fixed-parameter tractable*, ACM Transactions on Algorithms, 11 (2015), pp. 21:1–21:35. 2

[9] ———, *Chordal editing is fixed-parameter tractable*, Algorithmica, 75 (2016), pp. 118–137. 2, 3, 9

[10] J. Chen, I. A. Kanj, and W. Jia, *Vertex cover: further observations and further improvements*, Journal of Algorithms, 41 (2001), pp. 280–301. 2

[11] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer-Verlag, 2015. 2, 3, 4

[12] M. Cygan, L. Kowalik, and M. Pilipczuk, *Open problems from workshop on kernels, 2013*, URL: http://worker2013. mimuw. edu. pl/slides/worker-opl. pdf. 2

[13] H. Dell and D. van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, J. ACM, 61 (2014), p. 23. 2

[14] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Texts in Computer Science, Springer, 2013. 2, 3

[15] M. Farber, *On diameters and radii of bridged graphs*, Discrete Mathematics, 73 (1989), pp. 249–260. 5

[16] U. Feige, M. Hajiaghayi, and J. R. Lee, *Improved approximation algorithms for minimum weight vertex separators*, SIAM J. Comput., 38 (2008), pp. 629–657. 7

[17] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006. 2, 3

[18] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh, *Planar F-deletion: Approximation, kernelization and optimal FPT algorithms*, in FOCS, 2012. 1, 2

[19] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos, *Bidimensionality and kernels*, in Proc. 21st SODA, M. Charikar, ed., SIAM, 2010, pp. 503–510. 1

[20] F. V. Fomin, S. Saurabh, and Y. Villanger, *A polynomial kernel for proper interval vertex deletion*, SIAM J. Discrete Math., 27 (2013), pp. 1964–1976. 2

[21] L. Fortnow and R. Santhanam, *Infeasibility of instance compression and succinct PCPs for NP*, J. Comput. Syst. Sci., 77 (2011), pp. 91–106. 2

[22] T. Fujito, *A unified approximation algorithm for node-deletion problems*, Discrete Appl. Math., 86 (1998), pp. 213–231. 2

[23] A. C. Giannopoulou, B. M. P. Jansen, D. Lokshtanov, and S. Saurabh, *Uniform kernelization complexity of hitting forbidden minors*, CoRR, abs/1502.03965 (2015). to appear in ICALP 2015. 1, 2

[24] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980. 4

[25] P. Heggernes, P. van 't Hof, B. M. P. Jansen, S. Kratsch, and Y. Villanger, *Parameterized complexity of vertex deletion into perfect graph classes*, Theor. Comput. Sci., 511 (2013), pp. 172–180. 2

[26] B. M. P. Jansen, *Turing kernelization for finding long paths and cycles in restricted graph classes*, in Proc. 22th ESA, A. S. Schulz and D. Wagner, eds., vol. 8737 of Lecture Notes in Computer Science, Springer, 2014, pp. 579–591. 1

[27] B. M. P. Jansen and M. Pilipczuk, *Approximation and kernelization for chordal vertex deletion*, CoRR abs/1605.03001, (2016). 2, 3, 9

[28] Y. Ke, Y. Cao, X. Ouyang, and J. Wang, *Unit interval vertex deletion: Fewer vertices are relevant*, arXiv preprint arXiv:1607.01162, (2016). 2

[29] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar, *Linear kernels and single-exponential algorithms via protrusion decompositions*, ACM Trans. Algorithms, 12 (2016), p. 21. 1

[30] S. Kratsch, *Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP*, Algorithmica, 63 (2012), pp. 532–550. 1

[31] ———, *Recent developments in kernelization: A survey*, Bulletin of the EATCS, 113 (2014). 2, 3

[32] S. Kratsch and M. Wahlström, *Representative sets and irrelevant vertices: New tools for kernelization*, in Proc. 53rd FOCS, 2012, pp. 450–459. 1, 2

[33] S. Kratsch and M. Wahlström, *Compression via matroids: A randomized polynomial kernel for odd cycle transversal*, ACM Trans. Algorithms, 10 (2014), pp. 20:1–20:15. 1, 2

[34] T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, Journal of the ACM (JACM), 46 (1999), pp. 787–832. 7

[35] J. M. Lewis and M. Yannakakis, *The node-deletion problem for hereditary properties is NP-complete*, J. Comput. Syst. Sci., 20 (1980), pp. 219–230. 2

[36] D. Lokshtanov, N. Misra, and S. Saurabh, *Kernelization - preprocessing with a guarantee*, in The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday, vol. 7370 of Lecture Notes in Computer Science, Springer, 2012, pp. 129–161. 2, 3

[37] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, J. ACM, 41 (1994), pp. 960–981. 2

[38] D. Marx, *Chordal deletion is fixed-parameter tractable*, Algorithmica, 57 (2010), pp. 747–768. 2, 3

[39] D. Marx, B. O'Sullivan, and I. Razgon, *Finding small separators in linear time via treewidth reduction*, ACM Transactions on Algorithms, 9 (2013), p. 30. 2

[40] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, vol. 31 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006. 2, 3

[41] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen, *Network sparsification for steiner problems on planar and bounded-genus graphs*, in Proc. 55th FOCS, IEEE Computer Society, 2014, pp. 276–285. 1

[42] S. Thomassé, *A $4k^2$ kernel for feedback vertex set*, ACM Transactions on Algorithms, 6 (2010). 1, 2, 4

[43] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, *A new algorithm for generating all the maximal independent sets*, SIAM J. Comput., 6 (1977), pp. 505–517. 5

[44] V. V. Vazirani, *Approximation Algorithms*, Springer, 2001. 8

[45] M. Yannakakis, *Node-and edge-deletion NP-complete problems*, in Proceedings of the tenth annual ACM symposium on Theory of computing, STOC '78, New York, NY, USA, 1978, ACM, pp. 253–264. 2