Algorithmica (2022) 84:2335–2357
https://doi.org/10.1007/s00453-022-00971-7

Structural Parameterizations with Modulator Oblivion

Ashwin Jacob[1] · Fahad Panolan[2] · Venkatesh Raman[1] · Vibha Sahlot[3]

Received: 18 January 2021 / Accepted: 9 April 2022 / Published online: 3 May 2022

Abstract
It is known that problems like VERTEX COVER, FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL are polynomial time solvable in the class of chordal graphs. We consider these problems in a graph that has at most k vertices whose deletion results in a chordal graph when parameterized by k. While this investigation fits naturally into the recent trend of what is called 'structural parameterizations', here we assume that the deletion set is not given. One method to solve them is to compute a k-sized or an approximate ($f(k)$ sized, for a function f) chordal vertex deletion set and then use the structural properties of the graph to design an algorithm. This method leads to at least $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ running time when we use the known parameterized or approximation algorithms for finding a k-sized chordal deletion set on an n vertex graph. In this work, we design $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithms for these problems. Our algorithms do not compute a chordal vertex deletion set (or even an approximate solution). Instead, we construct a tree decomposition of the given graph in $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time where each bag is a union of four cliques and $\mathcal{O}(k)$ vertices. We then apply standard dynamic programming algorithms over this special tree decomposition. This special tree decomposition can be of independent interest. Our algorithms are, what are sometimes called *permissive* in the sense that given an integer k, they detect whether the graph has no chordal vertex deletion set of size at most k or output the special tree decomposition and solve the problem. We also show lower bounds for the problems we deal with under the strong exponential time hypothesis.

Keywords Parameterized complexity · Chordal graph · Tree decomposition · Strong exponential time hypothesis

A preliminary version appeared in the proceedings of IPEC 2020 [27].

✉ Ashwin Jacob
ajacob@imsc.res.in

[1] The Institute of Mathematical Sciences, HBNI, Chennai, India

[2] Department of Computer Science and Engineering, IIT Hyderabad, Hyderabad, India

[3] Computer Science Institute (CSI) of Charles University, 118 00, Prague, Czech Republic

1 Introduction and Motivation

The main motivation for parameterized complexity and algorithms is that hard problems have some parameters in their input, and feasible algorithms can be obtained when some of these parameters tend to be small. However, barring width parameters (like treewidth and cliquewidth), early parameterizations of problems were mostly in terms of solution size. However starting from the work of Fellows et al. [17] and Jansen et al. [18, 28], there has been a lot of study on parameterizations by some structure of the input. The motivations for these parameterizations are that many problems are computationally easy on special classes of graphs like edge-less graphs, forests, and interval graphs. Thus parameterizing by the size of a modulator (set of vertices in the graph whose removal results in a graph in easy graph class) became a natural choice of investigation. Examples of such parameterizations include CLIQUE and FEEDBACK VERTEX SET parameterized by the size of minimum vertex cover (i.e., modulator to edge-less graphs), VERTEX COVER parameterized by the size of minimum feedback vertex set (i.e., modulator to forests) [28, 29]. See also [35, 36] for more such parameterizations.

We continue this line of work on problems in input graphs that are not far from a chordal graph. By distance to a chordal graph, we mean the minimum number of vertices in the graph whose deletion results in a chordal graph. We call this set a chordal vertex deletion set (CVD). Specifically, we look at VERTEX COVER, FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL and some generalizations of these problems, parameterized by the size of a CVD, as these problems are polynomial time solvable in chordal graphs [10, 23, 40].

In problems for which the parameter is the size of a modulator, it is also assumed that the modulator is given with the input. This assumption can be removed if finding the modulator is also fixed-parameter tractable (FPT) parameterized by the modulator size. However, there are instances where finding the modulator is more expensive than solving the problem if the modulator is given. For example, finding a subset of k vertices whose deletion results in a perfect graph is known to be W-hard [25], whereas if the deletion set is given, then one can show (as explained a bit later in this section) that VERTEX COVER (thus INDEPENDENT SET) is FPT when parameterized by the size of the deletion set.

Hence Fellows et al. [18] ask whether INDEPENDENT SET (or equivalently, VERTEX COVER) is FPT when parameterized by a (promised) bound on the vertex-deletion distance to a perfect graph, without giving a minimum deletion set in the input. While we do not answer this question, we address a similar question in the context of problems parameterized by the distance to chordal graphs, another well-studied class of graphs where VERTEX COVER is polynomial time solvable whereas the best-known algorithm to find a k-sized chordal deletion set takes $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time [8]. We also do not know of a constant factor (FPT) approximation algorithm for CVD even with $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ running time. There are many recent results on polynomial time approximation algorithms for CHORDAL VERTEX DELETION [1, 31, 32] with the current best algorithm having a $\mathcal{O}(\log^2 \mathsf{opt})$ ratio, where opt is the size of minimum CVD [1]. If

we use this approximation algorithm and do branching (see Sect. 1.1 below), then we can obtain a $2^{\mathcal{O}(k \log^2 k)} n^{\mathcal{O}(1)}$ time algorithm for VERTEX COVER.

Hence, in a similar vein to the question by Fellows et al. we ask whether (minimum) VERTEX COVER (and other related problems) can be solved in $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time with only a promise on the size k of the chordal deletion set, and answer the question affirmatively. Our algorithms go one step further, not even needing the promise. They solve the problem or determine that the chordal deletion set is of size more than k.

Our Results Specifically, we give $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ algorithms for the problems defined below.

d- COLORABLE SUBGRAPH BY CVD	**Parameter:** k
Input: A graph $G = (V, E)$ and $k, \ell, d \in \mathbb{N}$.	
Question: Does there exist a vertex set X of size at most ℓ in G such that $G - X$ is d-colorable or output that minimum chordal vertex deletion set of G is of size more than k?	

When $d = 1$ and $d = 2$, the problem reduces to VERTEX COVER BY CVD and ODD CYCLE TRANSVERSAL BY CVD where we require the graph $G - X$ to be an independent set and bipartite, respectively. We also define FEEDBACK VERTEX SET BY CVD where we require the graph $G - X$ to be a forest.

We remark that our algorithms do not necessarily address the question of whether the input graph has a CVD of size at most k, and may solve the problem sometimes even when the CVD size is more than k.

We also show that all the problems mentioned above cannot be solved in $(2 - \epsilon)^k n^{\mathcal{O}(1)}$ time under Strong Exponential Time Hypothesis (SETH) even if a CVD of size k is given as part of the input. This matches the upper bound of the known algorithm for VERTEX COVER BY CVD when the modulator is given.

1.1 Related Work

When CVD is given If we are given a CVD S of size k along with an n-vertex graph G as the input, then one can easily get a $2^k n^{\mathcal{O}(1)}$ time algorithm (call it $\mathcal{A}$) for VERTEX COVER as follows. First, we guess the subset X of S that is part of our solution. Let Y be the subset of vertices in $V(G) \setminus S$ such that for each $y \in Y$ there is an edge between y and a vertex in $S \setminus X$. The set $X \cup Y$ is part of the VERTEX COVER solution and it will cover all the edges incident on S. Then we are left with finding an optimum vertex cover in $G - (S \cup Y)$ which is a chordal graph. This can be done in polynomial time. As we have 2^k choices for X, the total running time of the algorithm is $2^k n^{\mathcal{O}(1)}$. An FPT algorithm with $2^{O(k)} n^{\mathcal{O}(1)}$ time for FEEDBACK VERTEX SET BY CVD is given by Jansen et al [30] where they first find the modulator. This algorithm follows the algorithm to find a minimum feedback vertex set in bounded treewidth graphs. A similar trick works for ODD CYCLE TRANSVERSAL too when the modulator is given.

When the modulator is given, the FPT algorithms discussed above have been generalized for other problems and other classes of graphs (besides those that are k away from the class of chordal graphs). Let Φ be a Counting Monadic Second Order Logic (CMSO) formula and $t \geq 0$ be an integer. For a given graph $G = (V, E)$, the task

is to maximize $|X|$ subject to the following constraints: there is a set $F \subseteq V$ such that $X \subseteq F$, the subgraph $G[F]$ induced by F is of treewidth at most t, and structure $(G[F], X)$ models Φ. Note that the problem corresponds to finding a minimum vertex cover and a minimum feedback vertex set when $t = 0$ and $t = 1$ respectively when Φ is a tautology. For a polynomial $poly$, let G_{poly} be the class of graphs such that, for any $G \in G_{poly}$, graph G has at most $poly(n)$ minimal separators. Fomin et al. [21] gave a polynomial time algorithm for solving this optimization problem on the graph class G_{poly}. Consider $G_{poly} + kv$ to be the graph class formed from G_{poly} where to each graph we add at most k vertices of arbitrary adjacencies. Liedloff et al. [33] further proved that the above problem is FPT on $G_{poly} + kv$, with parameter k, where the modulator is also a part of the input. As a chordal graph has polynomially many minimal separators [23], we obtain that this problem parameterized by CVD size is FPT when the modulator is given.

Other 'Permissive' Problems Similar problems have been termed as 'permissive problems' in the context of testing satisfiability of CSPs (constraint satisfaction problems) with small-sized strong backdoors [22]. While detecting strong backdoors to a general CSP is hard, the authors address the question of satisfiability of CSPs where the backdoor set is not given, and the algorithm was supposed to solve satisfiability or determine that the backdoor set size is more than k.

An example line of work where a faster constant factor approximation algorithm is available is in the context of optimization problems parameterized by treewidth.

For example, the Independent Set problem parameterized by treewidth of the graph tw can be solved using standard dynamic programming (DP) in $2^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)} \cdot n$ time [12]. But the best known algorithm for outputting a tree-decomposition of minimum width takes time $\mathsf{tw}^{\mathcal{O}(\mathsf{tw}^3)}n$ [2]. Thus, the total running time is $\mathsf{tw}^{\mathcal{O}(\mathsf{tw}^3)}n$, when a tree decomposition is not given as an input. But one can overcome this by obtaining a tree decomposition of width $5\mathsf{tw}$ in time $2^{\mathcal{O}(\mathsf{tw})}n$ [4] and then applying the DP algorithm over the tree decomposition.

One previous example we know of a parameterized problem where the FPT algorithm solves the problem without the modulator or even the promise is Vertex Cover parameterized by the size of König Vertex Deletion set k. A König vertex deletion set of G is a subset of vertices of G whose removal results in a graph where the size of its minimum vertex cover and maximum matching is the same. In Vertex Cover by König Vertex Deletion, we are given graph $G = (V, E)$, $k, \ell \in \mathbb{N}$ and an assumption that there exists a König vertex deletion set of size k in G, here k is parameter. We want to ask whether there exists a vertex cover of size ℓ in G. Lokshtanov et al. [34] solve Vertex Cover by König Vertex Deletion in $1.5214^k n^{\mathcal{O}(1)}$ time without the promise.

Finally, we remark that there is an analogous line of work in the classical world of polynomial time algorithms. For example, it is known that finding a maximum clique in a unit disk graph is polynomial time solvable given a unit disk representation of the unit disk graph [9], though it is NP-hard to recognize whether a given graph is a unit disk graph [7]. Raghavan and Spinrad [38] give a permissive algorithm that given a graph either finds a maximum clique in the graph or outputs a certificate that the

given graph is not a unit disk graph. See also [6, 21, 24] for some other examples of permissive algorithms.

1.2 Our Techniques

The first step in our algorithms is to obtain, what we call a semi clique tree decomposition of the given graph if one exists. It is known [23] that every chordal graph has a clique-tree decomposition, i.e., a tree decomposition where every bag is a clique in the graph. If the modulator is given, then we can add it to each bag, and obtain a tree-decomposition where each bag is a clique plus at most k vertices. In our case (where the modulator is not given), we obtain a tree decomposition in $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time where each bag can be partitioned into $C \uplus N$, where C can be covered by at most 4 cliques in G and $|N| \leq 7k+5$. Here we also know a partition $C_1 \uplus C_2 \uplus C_3 \uplus C_4$ of C where each C_i is a clique. We call this tree decomposition a $(4, 7k+5)$-semi clique tree decomposition. Our result in this regard is formalized in the following theorem.

Theorem 1 *There is an algorithm that given a graph G and an integer k, runs in $\mathcal{O}(2^{7k} \cdot (kn^4 + n^{\omega+2}))$ time where ω is the matrix multiplication exponent, and either constructs a $(4, 7k+5)$-semi clique tree decomposition $\mathcal{T}$ of G or concludes that there is no chordal vertex deletion set of size k in G. Moreover, the algorithm also provides a partition $C_1 \uplus C_2 \uplus C_3 \uplus C_4 \uplus N$ of each bag of $\mathcal{T}$ such that $|N| \leq 7k+5$ and C_i is a clique in G for all $i \in \{1, 2, 3, 4\}$.*

After getting a $(4, 7k+5)$-semi clique tree decomposition, we then design DP algorithms for Vertex Cover, Feedback Vertex Set and Odd Cycle Transversal on this tree decomposition. Since the vertex cover of a clique has to contain all but one vertex of the clique, the number of ways the solution might intersect a bag of the tree is at most $\mathcal{O}(2^{7k}n^4)$. Using this fact, one can bound the running time for the DP algorithm for Vertex Cover by CVD to $\mathcal{O}(2^{7k}n^5)$. The overall running time would be the sum of the time taken to construct a $(4, 7k+5)$-semi clique tree decomposition and the time of the DP algorithm on this tree decomposition which is bounded by $\mathcal{O}(2^{7k}n^5)$. In the case of Feedback Vertex Set and Odd Cycle Transversal, again from each clique all but two vertices will be in the solution. Using this fact one can bound the running time of Feedback Vertex Set By CVD and Odd Cycle Transversal by CVD to be $2^{O(k)}n^{\mathcal{O}(1)}$.

Very recently, Fomin and Golovach [19] give subexponential algorithms to various problems on graphs which can be turned into a chordal graph by adding k edges. Similar to the line of work in this paper, they come up with an almost-clique tree decomposition (where each bag can be converted to a clique by adding k edges) and then apply dynamic programming algorithms on this tree decompositions. We use the dynamic programming algorithms in this paper on the tree decomposition we constructed to give algorithms for d- Colorable Subgraph parameterized by minimum CVD size.

Organization of the Paper In Sect. 2, we state the notations used in this paper and give the necessary preliminaries on tree decomposition and parameterized complexity. In Sect. 3, we prove Theorem 1. In Section 4, we first address d- Colorable Subgraph

BY CVD using dynamic programming on semi clique tree decomposition. We then give more direct and faster algorithms for VERTEX COVER BY CVD and ODD CYCLE TRANSVERSAL BY CVD and also for FEEDBACK VERTEX SET BY CVD. We then conclude this section with lower bounds on these problems assuming SETH.

2 Preliminaries

For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. We use $A \uplus B$ to denote the set formed from the union of disjoint sets A and B. For a function $w : X \to \mathbb{R}$, we use $w(D) = \sum_{x \in D} w(x)$.

We use the term graph for a simple undirected graph without loops and parallel edges. For a graph G, we use $V(G)$ and $E(G)$ to denote its vertex set and edge set, respectively. For $V' \subseteq V$, $G[V']$ and $G - V'$ denote the graph induced on V' and $V \setminus V'$, respectively. For a vertex $v \in V$, $G - v$ denotes the graph $G - \{v\}$. For a vertex $v \in V$, $N_G(v)$ and $N_G[v]$ denote the open neighborhood and closed neighborhood of v, respectively. That is, $N_G(v) = \{u : (v, u) \in E\}$ and $N_G[v] = N_G(v) \cup \{v\}$. Also we define for a subset $X \subseteq V(G)$, $N_G(X) = \bigcup_{v \in X}(N_G(v) \setminus X)$ and $N_G[X] = N_G(X) \cup X$. We omit the subscript G, when the graph is clear from the context.

A graph is *chordal* if it does not contain a cycle of length greater than or equal to 4 as an induced subgraph. A subset $S \subseteq V(G)$ such that $G - S$ is a chordal graph is called a *chordal vertex deletion set*. We say that a graph G is a union of ℓ cliques if $V(G) = V_1 \uplus \ldots \uplus V_\ell$ and V_i is a clique in G for all $i \in \{1, \ldots, \ell\}$. We use standard notation and terminology from the book [14] for graph-related terms which are not explicitly defined here.

A graph G is k-colorable if its vertices can be colored using k colors in such a way that the endpoints of every edge of G have two different colors.

Definition 1 (*Separator and separation*) Given a graph G and vertex subsets $A, B \subseteq V(G)$, a subset $C \subseteq V(G)$ is called a *separator* of A and B if every path from a vertex in A to a vertex in B (we call it $A - B$ path) contains a vertex from C. A pair of vertex subsets (A, B) is a *separation* in G if $A \cup B = V(G)$ and $A \cap B$ is a separator of $A \setminus B$ and $B \setminus A$.

Definition 2 (*Balanced separator and balanced separation*) For a graph G, a weight function $w : V(G) \to \mathbb{R}_{\geq 0}$ and $0 < \alpha < 1$, a set $S \subseteq V(G)$ is called an α-*balanced separator* of G with respect to w if for any connected component C of $G - S$, $w(V(C)) \leq \alpha \cdot w(V(G))$. A pair of vertex subsets (A, B) is an α-*balanced separation* in G with respect to w if (A, B) is a separation in G and $w(A \setminus B) \leq \alpha \cdot w(V(G))$ and $w(B \setminus A) \leq \alpha \cdot w(V(G))$.

Note that in the definition of balanced separator we only need a bound for the connected components of $G - S$ whereas in the case of balanced separation, we have a 2-partition $(A \setminus B, B \setminus A)$ of $V - S$ with $S = A \cap B$, each of which is bounded.

Definition 3 (*Tree decomposition*) A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree and for any $t \in V(T)$, a vertex subset $X_t \subseteq V(G)$ is associated with it, called a *bag*, such that the following conditions holds.

- $\bigcup_{t \in V(T)} X_t = V(G)$.
- For any edge $(u, v) \in E(G)$, there is a node $t \in V(T)$ such that $u, v \in X_t$.
- For any vertex $u \in V(G)$, the set $\{t \in V(T) : u \in X_t\}$ of nodes induces a connected subtree of T.

The width of the tree decomposition $\mathcal{T}$ is $\max_{t \in V(T)} |X_t| - 1$ and the *treewidth* of G is the minimum width over all tree decompositions of G.

Proposition 1 [15] *Let G be a graph and C be a clique in G. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree decomposition of G. Then, there is a node $t \in V(T)$ such that $C \subseteq X_t$.*

Definition 4 (*Clique tree decomposition*) A *clique tree decomposition* of a graph G is a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ where X_t is a clique in G for all $t \in V(T)$.

Proposition 2 *([23]) A graph is chordal if and only if it has a clique tree decomposition.*

Definition 5 A graph G is called a (c, ℓ)-*semi clique* if there is a partition $C \uplus N$ of $V(G)$ such that $G[C]$ is a union of at most c cliques and $|N| \leq \ell$.

Definition 6 ((c, ℓ)-*semi clique tree decomposition*) For a graph G and $c, \ell \in \mathbb{N}$, a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G is a (c, ℓ)-*semi clique tree decomposition* if $G[X_t]$ is a (c, ℓ)-semi clique for each $t \in V(T)$.

We define the Node Multiway Cut problem where we are given an input graph $G = (V, E)$, a set $T \subseteq V$ of terminals and an integer k. We want to ask whether there exists a set $X \subseteq V \setminus T$ of size at most k such that any path between two different terminals intersects X.

We use the following lemma in Sect. 3.

Proposition 3 [20] *Let T be a tree and $x, y, z \in V(T)$. Then there exists a vertex $v \in V(T)$ such that every connected component of $T - v$ has at most one vertex from $\{x, y, z\}$.*

Note that v can also be one of x, y or z in the above proposition.

SETH For $q \geq 3$, let δ_q be the infimum of the set of constants c for which there exists an algorithm solving q-SAT with n variables and m clauses in time $2^{cn} \cdot m^{\mathcal{O}(1)}$. The *strong exponential-time hypothesis* (SETH) conjectures that $\lim_{q \to \infty} \delta_q = 1$. SETH implies that CNF- SAT on n variables cannot be solved in $(2 - \epsilon)^n m^{\mathcal{O}(1)}$ time for any $\epsilon > 0$.

For definitions and notions on parameterized complexity, we refer to [12]. Throughout the paper, ω denotes the matrix multiplication exponent.

3 Semi Clique Tree Decomposition

Given a graph G and an integer k, we aim to construct a $(4, 7k + 5)$-semi clique tree decomposition $\mathcal{T}$ of G or conclude that G has no CVD of size at most k. We loosely follow the ideas used for the tree decomposition algorithm in [39] to construct a tree

decomposition of a graph G of width at most $4\mathsf{tw}(G)+4$, where $\mathsf{tw}(G)$ is the treewidth of G. But before that, we propose the following lemmas that we use in getting the required $(4, 7k + 5)$-semi clique tree decomposition.

Lemma 1 *Let G be a graph having a CVD of size k. Then G has a $(1, k)$-semi clique tree decomposition.*

Proof Let Y be the chordal vertex deletion set of G of size k. Since $G - Y$ is a chordal graph, it has a clique tree decomposition $\mathcal{T}'$. Adding Y to each bag of the tree decomposition $\mathcal{T}'$, we get a $(1, k)$-semi clique tree decomposition $\mathcal{T} = (T, \{X_t\}_{t\in V(T)})$ of G. □

Lemma 2 *For a graph G on n vertices with a CVD of size k, the number of maximal cliques in G is bounded by $\mathcal{O}(2^k \cdot n)$. Furthermore, there is an algorithm that given any graph G either concludes that there is no CVD of size k in G or enumerates all the maximal cliques of G in $\mathcal{O}(2^k \cdot n^{\omega+1})$ time where ω is the matrix multiplication exponent.*

Proof Let $X \subseteq V(G)$ be of size at most k such that $G - X$ is a chordal graph. For any maximal clique C in G let $C_X = C \cap X$ and $C_{G-X} = C \setminus X$. Since $G - X$ is a chordal graph, it has at most $n - k$ maximal cliques [23].

We claim that for a subset $C_X \subseteq X$ and a maximal clique Q in $G - X$, there is at most one subset $Q' \subseteq Q$ such that $C_X \cup Q'$ forms a maximal clique in G. If there are two distinct subsets Q_1, Q_2 of Q such that $C_X \cup Q_1$ and $C_X \cup Q_2$ are cliques in G, then $C_X \cup Q_1 \cup Q_2$ is a clique larger than the cliques $C_X \cup Q_1$ and $C_X \cup Q_2$. Thus, since there are at most 2^k subsets of X and at most n maximal cliques in G, the total number of maximal cliques in G is upper bounded by $2^k(n - k)$.

There is an algorithm that given a graph H, enumerates all the maximal cliques of H with $\mathcal{O}(|V(H)|^\omega)$ delay (the maximum time taken between outputting two consecutive solutions) [37]. If G has a CVD of size k, there are at most $2^k n$ maximal cliques in G. We use the algorithm to enumerate all the maximal cliques of the graph and keep the count of such cliques while doing so. If the count exceeds $2^k n$, we stop the algorithm concluding that the graph G has no CVD of size k. □

Lemma 3 *Let G be a graph having a CVD of size k and $w : V(G) \to \mathbb{R}_{\geq 0}$ be a weight function on $V(G)$. There exists a $\frac{2}{3}$-balanced separation (A, B) of G with respect to w such that the graph induced on the corresponding separator $G[A \cap B]$ is a $(1, k)$-semi clique.*

Proof First, we prove that there is a $\frac{1}{2}$-balanced separator X such that $G[X]$ is a $(1, k)$-semi clique. By Lemma 1, there is a $(1, k)$-semi clique tree decomposition $\mathcal{T} = (T, \{X_t\}_{t\in V(T)})$ of G. Arbitrarily root the tree of T at a node $r \in V(T)$. For any node $y \in V(T)$, let T_y denote the subtree of T rooted at node y and G_y denote the graph induced on the vertices of G present in the bags of nodes of T_y. That is $V(G_y) = \bigcup_{t\in V(T_y)} X_t$. Let t be the farthest node of T from the root r such that $w(V(G_t)) > \frac{1}{2}w(V(G))$. That is, for all nodes $t' \in V(T_t) \setminus \{t\}$, we have that $w(V(G_{t'})) \leq \frac{1}{2}w(V(G))$.

We claim that $X = X_t$ is a $\frac{1}{2}$-balanced separator of G. Let $t_1, \ldots, t_p$ be the children of t. Since X is a bag of the tree decomposition $\mathcal{T}$, each of the connected components of $G - X$ are contained either in $G_{t_i} - X$ for some $i \in [p]$ or $G[V(G) \setminus V(G_t)]$. Since $w(V(G_t)) > \frac{1}{2}w(V(G))$, we have $w(V(G) \setminus V(G_t)) < \frac{1}{2}w(V(G))$. By the choice of t, we have $w(V(G_{t_i})) \leq \frac{1}{2}w(V(G))$ for all $i \in [p]$.

Now we define a $\frac{2}{3}$-balanced separation (A, B) for G where the set $X = A \cap B$ is a $\frac{1}{2}$ balanced separator. Let $D_1, \ldots, D_q$ be the vertex sets of the connected components of $G - X$. Let $a_i = w(D_i)$ for all $i \in [q]$. Without loss of generality, assume that $a_1 \geq \ldots \geq a_q$. Let q' be the smallest index such that $\sum_{i=1}^{q'} a_i \geq \frac{1}{3}w(V(G))$ or $q' = q$ if no such index exists. Clearly, $\sum_{i=q'+1}^{q} a_i \leq \frac{2}{3}w(V(G))$. We prove that $\sum_{i=1}^{q'} a_i \leq \frac{2}{3}w(V(G))$. If $q' = 1$, $\sum_{i=1}^{q'} a_i = a_{q'} \leq \frac{1}{2}w(V(G))$ and we are done. Else, since q' is the smallest index such that $\sum_{i=1}^{q'} a_i \geq \frac{1}{3}w(V(G))$, we have $\sum_{i=1}^{q'-1} a_i < \frac{1}{3}w(V(G))$. We also note that $a_{q'} \leq a_{q'-1} \leq \sum_{i=1}^{q'-1} a_i < \frac{1}{3}w(V(G))$. Hence $\sum_{i=1}^{q'} a_i = \sum_{i=1}^{q'-1} a_i + a_{q'} \leq \frac{2}{3}w(V(G))$.

Now we define $A = X \cup \bigcup_{i \in [q']} D_i$ and $B = X \cup \bigcup_{i \in [q] \setminus [q']} D_i$. Notice that $X = A \cap B$ and (A, B) is a separation of G. Also notice that $w(A \setminus B) = \sum_{i=1}^{q'} a_i \leq \frac{2}{3}w(V)$ and $w(B \setminus A) = \sum_{i=q'+1}^{q} a_i \leq w(V(G)) - \frac{1}{3}w(V(G)) = \frac{2}{3}w(V(G))$ as $\sum_{i=1}^{q'} a_i \geq \frac{1}{3}w(V(G))$. Since X is a bag of the tree decomposition $\mathcal{T}$, $G[X]$ is a $(1, k)$-semi clique. □

Using Lemmas 2 and 3, we obtain the following corollary.

Corollary 1 *Let G be a graph with a CVD of size k. Let $N \subseteq V(G)$ with $5k+3 \leq |N| \leq 6k + 4$. Then there exists a partition (N_A, N_B) of N and a vertex subset $X \subseteq V(G)$ satisfying the following properties.*

- $|N_A|, |N_B| \leq 4k + 2$.
- *X is a vertex separator of N_A and N_B in the graph G.*
- *$G[X]$ is a $(1, k)$-semi clique.*

Moreover, there is an algorithm that given any graph G, either concludes that there is no CVD of size k in G or computes such a partition (N_A, N_B) of N and the set X in $\mathcal{O}(2^{7k} \cdot (kn^3 + n^{\omega+1}))$ time.

Proof Let us define a weight function $w : V(G) \to \mathbb{R}_{\geq 0}$ such that $w(v) = 1$ if $v \in N$ and 0 otherwise. From Lemma 3, we know that there exists a pair of vertex subsets (A, B) which is the balanced separation of G with respect to w where the graph induced on the corresponding separator $G[A \cap B]$ is a $(1, k)$-semi clique.

Let us define the partition (N_A, N_B). We add $(A \setminus B) \cap N$ to N_A and $(B \setminus A) \cap N$ to N_B. Since (A, B) is a balanced separation of G with respect to w, $|(A \setminus B) \cap N|$, $|(B \setminus A) \cap N| \leq \frac{2}{3}|N| \leq 4k + 2$. For each vertex $u \in (A \cap B) \cap N$, we iteratively add u to the currently smaller of the two sets of N_A and N_B. Since $|N| \leq 6k+4 \leq 2 \cdot (4k+2)$, we have $|N_A|, |N_B| \leq 4k + 2$ even after this process. This shows the existence of subsets N_A, N_B and $X = A \cap B$. But the proof is not constructive as the existence of

(A, B) uses the $(1, k)$-semi clique tree decomposition of G which requires the chordal vertex deletion set.

We now explain how to compute these subsets without the knowledge of a $(1, k)$-semi clique tree decomposition of G. Let $X = C'' \uplus N''$ where C'' is a clique and $|N''| \leq k$. We use Lemma 2 to either conclude that G has no CVD of size k or go over all maximal cliques of G to find a maximal clique D such that $C'' \subseteq D$. We can conclude that in the remaining graph $G[V \setminus D]$, there exists a separator $Z \subseteq N'' = X \setminus C''$ of size at most k for the sets N_A and N_B.

We go over all $2^{|N|} \leq 2^{6k+4}$ 2-partitions of N to guess the partition (N_A, N_B). Then we apply the classic Ford–Fulkerson maximum flow algorithm to find the separator Z of the sets N_A and N_B in the graph $G[V \setminus D]$. If $|Z| > k$, we can conclude that G has no CVD of size k in G. Thus, we obtained a set $X' = D \uplus Z$ such that $G[X']$ is a $(1, k)$-semi clique and X' is a vertex separator of N_A and N_B in the graph G.

Now we estimate the time taken to obtain these sets. We first go over all $2^k \cdot n$ maximal cliques of the graph which takes $\mathcal{O}(2^k \cdot n^{\omega+1})$ time. Then for each of the $2^k \cdot n$ maximal cliques, we go over at most 2^{6k+4} guesses for N_A and N_B. Finally, we use the Ford–Fulkerson maximum flow algorithm to find the separator of size at most k for N_A and N_B which takes $\mathcal{O}(k(n + m))$ time. Overall the running time is $\mathcal{O}(2^k \cdot n^{\omega+1} + (2^k n) \cdot 2^{6k} \cdot (k(n + m))) = \mathcal{O}(2^{7k} \cdot (kn^3 + n^{\omega+1}))$. □

Lemma 4 *Let G be a graph having a CVD of size k. Let C_1, C_2, C_3 be three distinct cliques in G. Then there exists a vertex subset $X \subseteq V(G)$ such that $G[X]$ is a $(1, k)$-semi clique and X is a separator of C_i and C_j for all $i, j \in \{1, 2, 3\}$ and $i \neq j$. Moreover, there is an algorithm that given any graph G, either concludes that there is no CVD of size k in G or computes X in $\mathcal{O}(4^k \cdot (kn^3 + n^{\omega+1}))$ time.*

Proof By Lemma 1, there is a $(1, k)$-semi clique tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G. By Proposition 1, we know that there exist nodes $t_1, t_2, t_3 \in V(T)$ such that $C_1 \subseteq X_{t_1}$, $C_2 \subseteq X_{t_2}$ and $C_3 \subseteq X_{t_3}$. If two of the three nodes t_1, t_2, t_3 is the same node t, then it can be easily seen that $X = X_t$ is the required separator as only at most one of C_1, C_2, and C_3 remains after its deletion.

Hence assume that all three nodes t_1, t_2, t_3 are distinct. From Proposition 3, we know that there exists a node $t \in V(T)$ such that (i) t_1, t_2 and t_3 are in different connected components of $T - t$. We claim that $X = X_t$ is the required separator. Since X is a bag in the $(1, k)$-semi clique tree decomposition $\mathcal{T}$, $G[X]$ is a $(1, k)$-semi clique. Because of statement (i), we have that X is a separator of C_i and C_j for all $i, j \in \{1, 2, 3\}$ and $i \neq j$. The proof is not constructive as we do not have a $(1, k)$-semi clique tree decomposition of G.

We compute a set X' such that $G[X']$ is a $(1, k)$-semi clique and X' is a separator of C_i and C_j for all $i, j \in \{1, 2, 3\}$ and $i \neq j$, without the knowledge of a $(1, k)$-semi clique tree decomposition of G. Let $X = C'' \uplus N''$ where C'' is a clique and $|N''| \leq k$. Using Lemma 2, we either conclude that G has no CVD of size k or we go over all the maximal cliques of the graph G. We know that $C'' \subseteq D$ for one of such maximal cliques D. Now in the graph $G[V \setminus D]$, we know that there exists a set $Z \subseteq N'' = X \setminus C''$ of size at most k which separates the cliques $C_x \setminus D$, $C_y \setminus D$ and $C_z \setminus D$. To find Z, we add three new vertices x', y' and z'. We make x' adjacent to all the vertices of $C_x \setminus D$, y' adjacent to all the vertices of $C_y \setminus D$ and z' adjacent to

all the vertices of $C_z \setminus D$. We find the node multiway cut Y of size at most k with the terminal set being $\{x', y', z'\}$. The set Y can be found in $\mathcal{O}(2^k km)$ using the known algorithm for node multiway cut [13, 26]. If the algorithm returns that there is no such set Y of size k, we conclude that there is no CVD of size at most k in G. Else we get a set $X' = D \uplus Y$ which satisfies the properties of X.

Now we estimate the time taken to obtain X'. We get all the $2^k \cdot n$ maximal cliques of the graph in $\mathcal{O}(2^k \cdot n^{\omega+1})$ time. Now for each maximal clique, we use the algorithm for node multiway cut with $\mathcal{O}(2^k km)$ running time. Thus, the overall running time is $\mathcal{O}(2^k \cdot n^{\omega+1} + (2^k n) \cdot (2^k km)) = \mathcal{O}(4^k \cdot (kn^3 + n^{\omega+1}))$. □

Now we prove our main result (i.e., Theorem 1) in this section. For convenience, we restate it here.

Theorem 1 *There is an algorithm that given a graph G and an integer k, runs in $\mathcal{O}(2^{7k} \cdot (kn^4 + n^{\omega+2}))$ time and either constructs a $(4, 7k + 5)$-semi clique tree decomposition $\mathcal{T}$ of G or concludes that there is no chordal vertex deletion set of size k in G. Moreover, the algorithm also provides a partition $C_1 \uplus C_2 \uplus C_3 \uplus C_4 \uplus N$ of each bag of $\mathcal{T}$ such that $|N| \leq 7k + 5$ and C_i is a clique in G for all $i \in \{1, 2, 3, 4\}$.*

Proof We assume that G is connected as if not we can construct a $(4, 7k + 5)$-semi clique tree decomposition for each connected component of G and attach all of them to a root node whose bag is empty to get the required $(4, 7k + 5)$-semi clique tree decomposition of G.

To construct a $(4, 7k + 5)$-semi clique tree decomposition $\mathcal{T}$, we define a recursive procedure Decompose(W, S, d) where $S \subset W \subseteq V(G)$ and $d \in \{0, 1, 2\}$. The procedure returns a rooted $(4, 7k + 5)$-semi clique tree decomposition of $G[W]$ such that S is contained in the root bag of the tree decomposition. The procedure works under the assumption that the following invariants are satisfied.

- $G[S]$ is a $(d, 6k + 4)$-semi clique and $W \setminus S \neq \emptyset$.
- $S = N_G(W \setminus S)$. Hence S is called the *boundary* of the graph $G[W]$.

To get the required $(4, 7k + 5)$-semi clique tree decomposition of G, we call Decompose$(V(G), \emptyset, 0)$ which satisfies all the above invariants. The procedure Decompose(W, S, d) calls procedures Decompose(W', S', d') and a new procedure SplitCliques(W', S') whenever $d = 2$. For these subprocedures, we will show that $|W' \setminus S'| < |W \setminus S|$. Hence by induction on the cardinality of $W \setminus S$, we will show the correctness of the Decompose procedure.

The procedure SplitCliques(W, S) with $S \subset W \subseteq V(G)$ also outputs a rooted $(4, 7k + 5)$-semi clique tree decomposition of $G[W]$ such that S is contained in the root bag of the tree decomposition. But the invariants under which it works are slightly different which we list below.

- $G[S]$ is a $(3, 5k + 3)$-semi clique and $W \setminus S \neq \emptyset$.
- $S = N_G(W \setminus S)$.

Notice that the only difference between invariants for Decompose and SplitCliques is the first invariant where we require $G[S]$ to be a $(3, 5k + 3)$-semi clique for SplitCliques and $(d, 6k + 4)$-semi clique for Decompose.

The procedure SplitCliques(W, S) calls procedures Decompose$(W', S', 2)$ where we will again show that $|W' \setminus S'| < |W \setminus S|$. Hence again by induction on the cardinality of $W \setminus S$, we will show the correctness. Now we describe how the procedure Decompose is implemented.

Implementation of Decompose(W, S, d): Notice that $d \in \{0, 1, 2\}$. Firstly, if $|W \setminus S| \leq k + 1$, we output the tree decomposition as a node r with bag $X_r = W$ and stop. Clearly, the graph $G[X_r]$ is a $(4, 7k + 5)$-semi clique and it contains S. Otherwise, we do the following.

We construct a set $\hat{S}$ with the following properties.

1. $S \subset \hat{S} \subseteq W \subseteq V(G)$.
2. $G[\hat{S}]$ is a $(d + 1, 7k + 5)$-semi clique. Let $\hat{S} = C' \uplus N'$ where $G[C']$ is the union of $d + 1$ cliques and $|N'| \leq 7k + 5$.
3. Every connected component of $G[W \setminus \hat{S}]$ is adjacent to at most $5k + 3$ vertices of N'.

Since $G[S]$ is a $(d, 6k + 4)$-semi clique, we have that $S = C \uplus N$, where $G[C]$ is the union of d cliques and $|N| \leq 6k + 4$.

Case 1 $|N| < 5k + 3$. We set $\hat{S} = S \cup \{u\}$, where u is an arbitrary vertex in $W \setminus S$. Note that this is possible as $W \setminus S \neq \emptyset$. Clearly $\hat{S}$ follows all the properties above.

Case 2 $5k + 3 \leq |N| \leq 6k + 4$. Note that $G[W]$ being a subgraph of G also has a chordal vertex deletion set of size at most k if G has it. Applying Corollary 1 for the graph $G[W]$ and the subset N, we either conclude that G has no CVD of size k or get a partition (N_A, N_B) of N, a subset $X \subseteq W$ and a partition $D \uplus Z$ of X, where D is a clique in $G[W]$ and $|Z| \leq k$, in time $\mathcal{O}(2^{7k} \cdot (kn^3 + n^{\omega+1}))$ such that $|N_A|, |N_B| \leq 4k + 2$ and X is a vertex separator of N_A and N_B in the graph $G[W]$. □

We define $\hat{S} = S \cup X \cup \{u\}$ where u is an arbitrary vertex in $W \setminus S$. We need to verify that $\hat{S}$ satisfies the required properties.

Claim 1 *The set $\hat{S}$ satisfies properties* (1), (2) *and* (3).

Proof Since $u \in W \setminus S$, $S \subset \hat{S}$. Hence $\hat{S}$ satisfies property (1).

We now show that $\hat{S}$ satisfies property (2). Recall that $S = C \uplus N$, where $G[C]$ is the union of d cliques and $|N| \leq 6k + 4$. We define sets $C' = C \cup D$ and $N' = ((N \cup Z) \setminus C') \cup \{u\}$ Notice that $\hat{S} = C' \cup N'$. Clearly $G[C']$ is the union of $d + 1$ cliques. Also $|N'| \leq |N| + |Z| + 1 \leq (6k + 4) + k + 1 \leq 7k + 5$. Thus $\hat{S}$ satisfies property (2).

We now show that $\hat{S}$ satisfies property (3). Recall $\hat{S} = C' \cup N'$, where $C' = C \cup D$ and $N' = ((N \cup Z) \setminus C') \cup \{u\}$. Recall that $X = D \cup Z \subseteq \hat{S}$ is separator of N_A and N_B where $N = N_A \uplus N_B$ and $|N_A|, |N_B| \leq 4k + 2$. This implies that any connected component H in $G[W \setminus X]$ can contain at most $4k + 2$ vertices from N as the neighborhood of $V(H)$ is contained in X, because X is a separator. Moreover $|Z| \leq k$. This implies that any connected component in $G[W \setminus \hat{S}]$ is adjacent to at most $4k+2$ vertices in N and at most k vertices in Z, and hence at most $5k+3$ vertices in $N' = ((N \cup Z) \setminus C') \cup \{u\}$. □

Now we define the recursive subproblems arising in the procedure Decompose (W, S, d) using the constructed set $\hat{S}$. If $\hat{S} = W$, then there will not be any recursive subproblem. Otherwise, let $P_1, P_2, \ldots, P_q$ be vertex sets of the connected components of $G[W \setminus \hat{S}]$ and $q \geq 1$ because $\hat{S} \neq W$. We have the following cases:

Case 1 $d < 2$: For each $i \in [q]$, recursively call the procedure Decompose$(W' = N_G[P_i], S' = N_G(P_i), d + 1)$.

We now show that the invariants are satisfied for procedures Decompose$(W' = N_G[P_i], S' = N_G(P_i), d + 1)$ for all $i \in [q]$. We start by noticing that since $d < 2$, $d + 1 \leq 2$ which is required for the validity of the procedure. Let $Q_i = S' \cap N'$. Note that from condition (3) for $\hat{S}$, we have $|Q_i| \leq 5k + 3$. Since $S' \setminus Q_i \subseteq C'$ and $G[C']$ is a union of $d + 1$ cliques, $G[S']$ forms a $(d + 1, 5k + 3)$-semi clique which is also a $(d + 1, 6k + 4)$-semi clique. Also by definition of neighbourhoods, $P_i = N_G[P_i] \setminus N_G(P_i) = W' \setminus S'$. Since P_i is a non-empty set by definition, $W' \setminus S'$ is non-empty. Hence the first invariant required for the Decompose is satisfied. Since $S' = N_G(P_i) = N_G(N_G[P_i] \setminus N_G(P_i)) = N_G(W' \setminus S')$, the second invariant is satisfied.

Case 2 $d = 2$: For each $i \in [q]$, recursively call the procedure SplitCliques$(W' = N_G[P_i], S' = N_G(P_i))$. We can show that the invariants for SplitCliques are satisfied with the proofs similar to the previous case.

We now explain how to construct the $(4, 7k + 5)$-semi clique tree decomposition using Decompose(W, S, d). Here, we assume that Decompose$(W', S', d + 1)$ and SplitCliques(W', S') return a $(4, 7k + 5)$-semi clique tree decomposition $G[W']$ when $|W' \setminus S'| < |W \setminus S|$. That is, we apply induction on $|W \setminus S|$. Look at the subprocedures Decompose(W', S', d) and SplitCliques(W', S'). We have $W' \setminus S' = N_G[P_i] \setminus N_G(P_i) = P_i$ which is a subset of $W \setminus \hat{S}$ which in turn is a strict subset of $W \setminus S$. Hence $|W' \setminus S'| < |W \setminus S|$. Hence we apply induction on $|W \setminus S|$ to the subprocedures. Let $\mathcal{T}_i$ be the $(4, 7k+5)$-semi clique tree decomposition obtained from the subprocedure with $W' = N_G[P_i]$ and $S' = N_G(P_i)$. Let r_i be the root of $\mathcal{T}_i$ whose associated bag is X_{r_i}. By induction hypothesis $S' \subseteq X_{r_i}$. We create a node r with the corresponding bag $X_r = \hat{S}$. For each $i \in [q]$, we attach $\mathcal{T}_i$ to r by adding edge (r, r_i). Let us call the tree decomposition obtained so with root r as $\mathcal{T}$. We return $\mathcal{T}$ as the output of Decompose(W, S, d). By construction, it easily follows that $\mathcal{T}$ is a $(4, 7k + 5)$-semi clique tree decomposition of the graph $G[W]$ with the root bag containing S. We note that when $W = \hat{S}$, the procedure returns a single node tree decomposition with $X_r = W = \hat{S}$.

Implementation of SplitCliques **Procedure:** Again if $|W \setminus S| \leq k + 1$, we output the tree decomposition as a node r with bag $X_r = W$ and stop. Clearly the graph $G[X_r]$ is a $(4, 7k + 5)$-semi clique and it contains S. Otherwise we do the following. Let $S = C \uplus N = (C_x \uplus C_y \uplus C_z) \uplus N$ where C_x, C_y and C_z are the vertex sets of the three cliques in $G[C]$. We apply Lemma 4 to graph $G[W]$ and sets C_x, C_y and C_z, to either conclude that G has no CVD of size k or obtain a set Y such that Y separates the sets C_x, C_y and C_z and $G[Y]$ is a $(1, k)$-semi clique. Let $Y = D \uplus X$ where D is a clique and $|X| \leq k$.

Let $Y' = Y \cup \{u\}$ where u is any arbitrary vertex from $W \setminus S$ which we know to be non-empty. If $S \cup Y' = W$, then it will not call any recursive subproblem. Otherwise,

let $P_1, P_2, \ldots, P_q$ be the connected components of the graph $G[W \setminus (S \cup Y')]$. We recursively call Decompose$(W' = N_G[P_i], S' = N_G(P_i), 2)$ for all $i \in [q]$.

Since Y' is a separator of the cliques C_x, C_y and C_z, any connected component P_i will have neighbours to at most one of the three cliques $C_x \setminus Y'$, $C_y \setminus Y'$ and $C_z \setminus Y'$ in $G[W \setminus (S \cup Y')]$. We show that the invariants required for the procedure Decompose are satisfied in these subproblems. Let us focus on the procedure Decompose$(W' = N_G[P_i], S' = N_G(P_i), 2)$ which has neighbours only to the set $C_x \setminus Y'$. We define sets $C' = C_x \cup D$ and $N' = (N \cup X \cup \{u\}) \setminus C'$. The vertex set P_i has neighbours only to the set $(C_x \uplus N) \cup Y' = (C_x \uplus N) \cup (D \uplus X) \cup \{u\} = (C_x \cup D) \cup (N \cup X \cup \{u\}) = C' \uplus N'$. Clearly $G[C']$ is the union of at most two cliques and $|N'| \leq |N| + |X| + 1 = 5k + 3 + k + 1 \leq 6k + 4$. Hence the first invariant is satisfied for the procedure Decompose$(N_G[P_i], N_G(P_i), 2)$. The proof of the second invariant is the same as that of the subproblems of Decompose procedure. The satisfiability of invariants for other subprocedures can also be proven similarly.

We now construct the $(4, 7k + 5)$-semi clique tree decomposition returned by SplitCliques (W, S). Again we apply induction on $|W \setminus S|$. Consider the subprocedures Decompose(W', S', d). We have $W' \setminus S' = N_G[P_i] \setminus N_G(P_i) = P_i$ which is a subset of $W \setminus (S \cup Y')$ which in turn is a strict subset of $W \setminus S$ as $u \in W \setminus S$ is present in Y'. Hence $|W' \setminus S'| < |W \setminus S|$ and we apply induction on $|W \setminus S|$ to the subprocedures. Let $\mathcal{T}_i$ be the $(4, 7k + 5)$-semi clique tree decomposition obtained from the subprocedure with $W' = N_G[P_i]$ and $S' = N_G(P_i)$. Let r_i be the root of $\mathcal{T}_i$ whose bag X_{r_i} we show contains S'. We create a node r with the corresponding bag $X_r = S \cup Y' = (C_x \uplus C_y \uplus C_z \uplus D) \uplus N'$. For each $i \in [q]$, we attach $\mathcal{T}_i$ to r by adding edge (r, r_i). Let us call the tree decomposition obtained so with root r as $\mathcal{T}$. We return $\mathcal{T}$ as the output of SplitCliques(W, S, d). By construction, it easily follows that $\mathcal{T}$ is a $(4, 7k + 5)$-semi clique tree decomposition of the graph $G[W]$ with the root bag containing S. We mention that when $W = S \cup Y'$, the procedure returns a single node tree decomposition with $X_r = W$.

Running time analysis In the procedure Decompose, we invoke Corollary 1 which takes $\mathcal{O}(2^{7k} \cdot (kn^3 + n^{\omega+1}))$ time. For the procedure SplitCliques, we invoke Lemma 4 which takes $\mathcal{O}(4^k \cdot (kn^3 + n^{\omega+1}))$ time. All that is left is to bound the number of calls of the procedures Decompose and SplitCliques. Each time Decompose or SplitCliques is called, it creates a set $\hat{S}$ (in the case of SplitCliques, $\hat{S} = S \cup Y'$) which is a strict superset of S. This allows us to map each call of Decompose or SplitCliques to a unique vertex $u \in \hat{S} \setminus S$ of $V(G)$. Hence the total number of calls of Decompose and SplitCliques is not more than the total number of vertices n. Hence the overall running time of the algorithm which constructs the $(4, 7k+5)$-semi clique tree decomposition of G is $\mathcal{O}(2^{7k} \cdot (kn^4 + n^{\omega+2}))$. □

Faster Algorithm[1] We can get a faster algorithm by making use of the fact that any C_4-free graph has $O(n^2)$ maximal cliques [16]. The algorithm first repeatedly find induced C_4s if any and delete the vertices of C_4 from G. If there are more than k disjoint induced C_4s found, then return that G has no CVD of size at most k. Let Z denote the union of the vertices removed in this process. We now apply the algorithm

[1] We acknowledge Bart Jansen for the ideas leading to this algorithm.

in Theorem 1 on the graph $G - Z$ which is C_4-free. Note that the graph $G - Z$ has $O(n^2)$ maximal cliques. This drops the $2^k n$ factor to n^2 in the running times of the algorithms of Corollary 1 and 4 invoked in the algorithm of Theorem 1. Hence the running time to obtain the semi clique tree decomposition $\mathcal{T}'$ of $G - Z$ drops to $\mathcal{O}(2^{6k} \cdot (kn^5 + n^{\omega+2}))$. We now obtain a semi clique tree decomposition $\mathcal{T}$ of G from $\mathcal{T}'$ by adding Z to every bag of $\mathcal{T}'$.

We now prove that $\mathcal{T}$ is a $(4, 7k + 5)$-semi clique tree decomposition. Let $k_1 \leq k$ denote the number of disjoint C_4's present in Z. Hence $|Z| = 4k_1$. Since Z has k_1 disjoint C_4's, any CVD of G contains at least k_1 vertices of Z. Hence G has a CVD of size k if and only if $G-Z$ has a CVD of size $k-k_1$. The algorithm of Theorem 1 either concludes that $G - Z$ has no CVD of size $k - k_1$ or returns a $(4, 7(k - k_1) + 5)$-semi clique tree decomposition $\mathcal{T}'$ of $G - Z$. In the former case we can correctly conclude that G has no CVD of size k. In the later case, we now obtain a semi clique tree decomposition $\mathcal{T}$ of G from $\mathcal{T}'$ by adding Z to every bag of $\mathcal{T}'$. The vertices which are not part of the cliques in every bag of $\mathcal{T}$ is $7(k-k_1)+5+|Z| = 7(k-k_1)+5+4k_1$ which is at most $7k + 5$. Hence $\mathcal{T}$ is a $(4, 7k + 5)$-semi clique tree decomposition. Hence we have the following theorem.

Theorem 2 *There is an algorithm that given a graph G and an integer k runs in time $\mathcal{O}(2^{6k} \cdot (kn^5 + n^{\omega+2}))$ and either constructs a $(4, 7k+5)$-semi clique tree decomposition $\mathcal{T}$ of G or concludes that there is no chordal vertex deletion set of size k in G. Moreover, the algorithm also provides a partition $C_1 \uplus C_2 \uplus C_3 \uplus C_4 \uplus N$ of each bag of $\mathcal{T}$ such that $|N| \leq 7k + 5$ and C_i is a clique in G for all $i \in \{1, 2, 3, 4\}$.*

We note that though the above algorithm is faster, it does not improve the running time of algorithms of the dynamic programming algorithms in Section 4. This is because these algorithms store solutions for every possible subset of the non-clique part of the bags which is at least 2^{7k}. Hence in the following section, we continue using the algorithm in Theorem 1.

4 Structural Parameterizations with Chordal Vertex Deletion Set

Theorem 3 d-Colorable Subgraph by CVD *can be solved in $d^{4d+7k+5} 2^{3(7k+5)} n^{\mathcal{O}(d)}$ time.*

Proof First, we use Theorem 1 to construct a $(4, 7k+5)$-semi clique tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G in $2^{7k} n^{\mathcal{O}(1)}$ time. Now we use the dynamic programming algorithm on tree decompositions given by Fomin and Golovach (Theorem 1 of [19]) on $\mathcal{T}$ to find the maximum sized induced subgraph H of G such that H is d-colorable. Note that the set $V(G) \setminus V(H)$ is the solution that we are looking for and if its size is at most ℓ, we return YES. Else we return NO.

This dynamic programming algorithm defines a state $cost(t, S, c)$ for all nodes $t \in V(T)$, subsets $S \subseteq X_t$ such that $G[S]$ is d-colorable and a function $c : S \to [d]$. Since each bag X_t of $\mathcal{T}$ is a $(4, 7k + 5)$ semi clique, at most d vertices of each clique can be part of S as else there is a presence of a $(d + 1)$ sized clique in S which is not d-colorable. Hence we can bound the size of S as $4d + 7k + 5$ and also bound the

number of possible subsets S as $n^{4d}2^{7k+5}$. Number of possible functions c is at most $d^{|S|}$ which is at most $d^{4d+7k+5}$. Hence we bound the number of states $cost(t, S, c)$ as $d^{4d+7k+5}2^{7k+5}n^{\mathcal{O}(d)}$. For each state, the time taken is $\mathcal{O}(|S|^2)$. Hence the overall running time is $d^{4d+7k+5}2^{3(7k+5)}n^{\mathcal{O}(d)}$. □

Corollary 2 VERTEX COVER BY CVD *and* ODD CYCLE TRANSVERAL BY CVD *can be solved in* $2^{21k}n^{\mathcal{O}(1)}$ *and* $2^{28k}n^{\mathcal{O}(1)}$ *time, respectively.*

We can directly use the dynamic programming on bounded treewidth to get algorithms with better running times for VERTEX COVER BY CVD and ODD CYCLE TRANSVERSAL BY CVD and for FEEDBACK VERTEX SET BY CVD using the fact that any vertex cover contains all but one from each clique and any odd cycle transversal and feedback vertex set contains all but two from each clique.

Theorem 4 *Given a graph G and an integer k, there exist algorithms that determine that G has no CVD of size k or*

- *find a minimum vertex cover in* $2^{7k}n^{O(1)}$ *time, and*
- *find a minimum odd cycle transversal in* $3^{7k}n^{\mathcal{O}(1)}$ *time, and*
- *find a minimum feedback vertex set in* $2^{\omega 7k}n^{\mathcal{O}(1)}$ *time.*

Proof First, we use Theorem 1 to construct a $(4, 7k+5)$-semi clique tree decomposition $\mathcal{T} = (T, \{X_t\}_{t\in V(T)})$ of G in $2^{7k}n^{\mathcal{O}(1)}$ time. Arbitrarily root the tree T at a node r. Let $X_t = C_{t,1} \uplus \ldots \uplus C_{t,4} \uplus N_t$ where $|N_t| \leq 7k + 5$ and $C_{t,j}$ is a clique in G for all $j \in \{1, \ldots, 4\}$. In the tree decomposition $\mathcal{T} = (T, \{X_t\}_{t\in V(T)})$, for any vertex $t \in V(T)$, we call D_t to be the set of vertices that are descendant of t. We define G_t to be the subgraph of G on the vertex set $X_t \cup \bigcup_{t'\in D_t} X_{t'}$.

Proof sketch of the algorithm for VERTEX COVER BY CVD**:** We briefly explain the dynamic programming (DP) table entries on $\mathcal{T}$. In a standard DP for each node $t \in V(T)$ and $Y \subseteq X_t$, we have a table entry $DP[Y, t]$ which stores the size of a minimum vertex cover S of G_t such that $Y = X_t \cap S$ and if no such vertex cover exists, then $DP[Y, t]$ stores ∞. In fact, we only need to store $DP[Y, t]$ whenever it is not equal to ∞. Now consider a bag X_t in $\mathcal{T}$. For any $Y \subseteq X_t$, if $|C_{t,j} \setminus Y| \geq 2$ for any $j \in [4]$, then $DP[Y, t] = \infty$ because $C_{t,j}$ is a clique. Therefore, we only need to consider subsets $Y \subseteq X_t$ for which $|C_{t_j} \setminus Y| \leq 1$ for all $j \in [4]$. The number of choices of such subsets Y is bounded by $\mathcal{O}(2^{7k}n^4)$. This implies that the total number of DP table entries is $\mathcal{O}(2^{7k}n^5)$. All these values can be computed in time $\mathcal{O}(2^{7k}n^{O(1)})$ time using standard dynamic programming in a bottom up fashion. For more details about dynamic programming over tree decomposition, see [12].

Proof sketch of the algorithm for ODD CYCLE TRANSVERSAL BY CVD: Any odd cycle transversal contains all but at most two vertices from each clique $C_{1,j}$, $i \in [4]$. Using this fact we can bound the number of DP table entries to be at most $3^{7k}n^{\mathcal{O}(1)}$. Then, by computing the entries in a bottom-up fashion in time $3^{7k}n^{\mathcal{O}(1)}$ using standard arguments.

Proof sketch of algorithm for FEEDBACK VERTEX SET BY CVD: We use the ideas from the DP algorithm for FEEDBACK VERTEX SET using the rank-based approach [3].

We give a more detailed algorithm as the techniques for solving FEEDBACK VERTEX SET parameterized by treewidth are slightly sophisticated and it may not be obvious for the reader that these techniques extend to semi-clique tree decompositions.

We create an auxiliary graph G' by adding a vertex v_0 to G and making it adjacent to all the vertices of G. Let E_0 be the set of newly added edges. Thus we add v_0 to all the bags to get the tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G'. We use a dynamic programming algorithm for FEEDBACK VERTEX SET on $\mathcal{T}$ where the number of entries of the DP table we will show to be $2^{7k+5}n^{11}$. Let $X_t = C_{t,1} \uplus \ldots \uplus C_{t,4} \uplus N_t$ for all $t \in V(T)$ where $|N_t| \leq 7k + 5$ and $C_{t,j}$ is a clique in G for all $j \in \{1, \ldots, 4\}$. For a node $t \in V(T)$, a subset $Y \subseteq X_t$ and integers $i, j \in [n]$, we define the entry $DP[t, Y, i, j]$. The entry $DP[t, Y, i, j]$ stores a partition $\mathcal{P}$ of Y if

- there exists a vertex subset $X \subseteq D_t$, $v_0 \in X$ such that $X \cap X_t = Y$ and
- there exists an edge subset $X_0 \subseteq E(G_t) \cap E_0$ such that in the graph $(X, E(G_t[X \setminus \{v_0\}]) \cup X_0)$, we have i vertices, j edges, no connected component is fully contained in $D_t \setminus X_t$ and the elements of Y are connected according to the partition $\mathcal{P}$.

We set $DP[t, Y, i, j] = \infty$ if the entry can be inferred to be invalid from Y.

We claim that the FEEDBACK VERTEX SET BY CVD instance (G, k) is a yes instance if and only if for the root r of $\mathcal{T}$ with $X_r = \{v_0\}$ and some $i \geq |V| - \ell$, we have $DP[r, \{v_0\}, i, i-1]$ to be non-empty. In the forward direction, we have a feedback vertex set W of size ℓ. The graph $G - W$ has $|V| - \ell$ vertices and $|V| - \ell - c$ edges where c is the number of connected components of $G - W$. We define $X = V \setminus W \cup \{v_0\}$ and X_0 to be c edges connecting v_0 to any one of the vertices of each of the c components of $V \setminus W$. We have $|X| \geq |V| - \ell$. The graph $(X, E(G_t[X \setminus \{v_0\}]) \cup X_0)$ has $|V| - \ell$ edges and satisfies the properties required for an entry in $DP[r, \{v_0\}, i, i - 1]$. In the reverse direction, we have a graph $(X, E(G_t[X \setminus \{v_0\}]) \cup X_0)$ having i edges and $i - 1$ edges. Since no connected component of the graph can be contained in $V(G_t) \setminus \{v_0\}$, the graph is a tree. Hence $V \setminus X$ is a feedback vertex set.

Now we give the recurrence relations for computing $DP[t, Y, i, j]$. For this purpose, we convert $\mathcal{T}$ into a nice tree decomposition which can be done in $\mathcal{O}(n^3)$ time. Since each bag contains cliques of size $\mathcal{O}(n)$, the number of nodes of $\mathcal{T}$ can also blow up to be $\mathcal{O}(n^3)$ with $\mathcal{O}(n^2)$ new nodes possibly added for each and edge of $\mathcal{T}$ corresponding to $\mathcal{O}(n^2)$ edges and vertices added or removed to obtain the collection of the cliques in the child node from the collection in the parent node.

The recurrences for computing $DP[t, Y, i, j]$ more or less remains the same as in [3]. Before we state them, we define some operations on a family $\mathcal{A}$ of partitions of a universe U.

- Union: For two families $\mathcal{A}$ and $\mathcal{B}$ of partitions of U, we define the union $\mathcal{A} \cup \mathcal{B}$ as the family obtained by taking the union of both families.
- Insert: For a family of partitions $\mathcal{A}$ and set X such that $X \cap U = \phi$, $insert(X, \mathcal{A})$ is the family of partitions obtained by adding each element of X as singleton sets in each of the partitions of $\mathcal{A}$.
- Glue: For elements u, v, $glue(uv, \mathcal{A})$ is obtained by combining the sets containing u and v in each of the partitions of $\mathcal{A}$.

- Project: For a family of partitions $\mathcal{A}$ and set $X \subseteq U$, $project(X, \mathcal{A})$ is the family obtained by removing all the elements of X from each of the partitions, but discarding the partition if doing so reduces the number of sets in the partition.
- Join: For a partition P of universe U and Q of universe U', the join of P and Q is defined as follows. Look at a graph G over vertices $U \cup U'$. Look at each set S in P and turn the corresponding vertex set into a clique. We do so for all the sets in P as well as Q. Now, look at the set of connected components of G. We get a partition of $U \cup U'$ with each set of the partition being the vertex set of the corresponding connected component. This partition is called the join of P and Q. For a family of partitions $\mathcal{A}$ over U and a family of partitions $\mathcal{B}$ over U', $join(\mathcal{A}, \mathcal{B})$ is the family of partitions over $U \cup U'$ obtained by taking the join of each pair of partitions from $\mathcal{A}$ and $\mathcal{B}$.

We have the following recurrence relations.

- Leaf Node: We set the entry $DP[t, \phi, 0, 0] = \{\phi\}$ and all other entries as invalid.
- Introduce vertex Node: Let t and t' be the parent and child nodes with vertex v being introduced in t. We have

$$DP[t, Y, i, j] = \begin{cases} \infty & \text{if } v = v_0 \text{ and } v \notin Y \\ insert(v, DP[t', Y \setminus \{v\}, i-1, j]) & \text{if } v \in Y \\ DP[t', Y, i, j] & \text{otherwise} \end{cases}$$

The first case is to ensure that if the vertex introduced is v_0, then it has to be in Y. Otherwise, if $v \in Y$, we extend solutions that do not contain v with $i-1$ vertices by adding singleton v to each of the partitions.

- Forget vertex node: Let t and t' be the parent and child nodes with vertex v being forgotten in t. We have

$$DP[t, Y, i, j] = DP[t', Y, i, j] \cup project(v, DP[t', Y \cup \{v\}, i, j])$$

We extend solutions of child node t' for both the cases when v is present or absent in the corresponding set Y. We do so by taking the union of partitions for both cases. In the case when v is present, we make sure that the partitions where v is a singleton are not added. This is because the corresponding component can no longer be connected as it has no intersection with X_t.

- Introduce Edge Node: Let t and t' be the parent and child nodes with edge (u, v) being added in t. We have

$$DP[t, Y, i, j] = \begin{cases} DP[t', Y, i, j] \cup glue(v_0 v, DP[t', Y, i, j-1]) & \text{if } u = v_0 \text{ and } v \in Y \\ DP[t', Y, i, j] \cup glue(v_0 u, DP[t', Y, i, j-1]) & \text{if } v = v_0 \text{ and } u \in Y \\ glue(uv, DP[t', Y, i, j-1]) & \text{if } u, v \in Y \\ DP[t', Y, i, j] & \text{otherwise} \end{cases}$$

If $u = v_0$ and $v \in Y$ (or the symmetric case), then the edge v_0v may or may not be part of the maximal induced forest corresponding to the solution. Hence we can choose to insert v_0v or not. Else if both u and v are present in Y, then edge (u, v) has to be present in the maximal induced forest. In the cases where the edge is present, we obtain the corresponding family of partitions by gluing sets containing u and v for each partition.

- Join node: Let t be the parent node and t', t'' be the child nodes. We have

$$DP[t, Y, i, j] = \bigcup_{i_1+i_2=i-|Y|, j_1+j_2=j} join(DP[t', Y, i_1, j_1], DP[t'', Y, i_2, j_2])$$

A pair of vertices x, y in X_t is connected in G_t if there is a vertex $z \in X_t$ such that x and z is connected in $G_{t'}$ and y and z is connected in $G_{t''}$. The partition of Y corresponding to this connectivity is obtained exactly via the join of pair of partitions in the entries of t' and t''.

Now we bound the number of table entries and the number of partitions stored for each entry. Consider a bag X_t in $\mathcal{T}$. For any $Z \subseteq X_t$, We focus on the sets $C_{t,j} \setminus Z$ which is part of the forest. If $|C_{t,j} \setminus Z| \geq 3$ for any $j \in [4]$, then $DP[t, Y, i, j] = \infty$ because $G[C_{t,j}]$ contains a triangle as $C_{t,j}$ is a clique. Therefore, we only need to consider subsets $Z \subseteq X_t$ for which $|C_{t,j} \setminus Z| \leq 2$ for all $j \in [4]$. Hence we have $|C_i| \leq 2$. The number of choices for each C_i is at most $|C_{t,i}| \leq n^2$. We also have $|Y| \leq |N_t| \leq 7k + 5$. Since the number of nodes of $\mathcal{T}$ is $\mathcal{O}(n^3)$, we have the total number of DP table entries is $\mathcal{O}(2^{7k}n^{13})$. In each DP table entry $DP[t, Y, i, j]$, we store partitions of Y. The cardinality of Y is bounded by $7k+13$ as $|C_{t_j} \setminus Y| \leq 2$ for all $j \in [4]$. Hence the number of partitions stored in a particular entry $DP[t, Y, i, j]$ can be as huge as $|Y|^{\mathcal{O}(|Y|)}$ which is bounded by $(7k + 13)^{O(k)}$. But as we will see below, we devise a reducing routine that allows us to only store at most 2^{7k+13} partitions in each table entry.

We use the ideas from [3] to bound the time taken to compute all the table entries of a particular node t. In particular, for each table enrty $DP[t, Y, i, j]$, after obtaining a family $\mathcal{A}$ of partitions over a universe U using recurrence relations, we use reducing algorithm Theorem 3.7 of [3] to obtain a subfamily $\mathcal{A}'$ of size $2^{|U|}$ which "represents" $\mathcal{A}$. Since the subfamily $\mathcal{A}'$ represents $\mathcal{A}$, it can be used for furthur evaluations in the dynamic programming algorithm. For more details, we refer to [3].

Let us first bound the time taken to compute the recurrence relations. Using Proposition 3.3 of [3], given to families $\mathcal{A}$ and $\mathcal{B}$ over a universe U, the time taken for every operation is bounded by $|\mathcal{A}||\mathcal{B}||U|^{\mathcal{O}(1)}$. The leading factor is the time taken for the reducing algorithm Theorem 3.7 of [3] which is bounded by $\mathcal{O}(|\mathcal{A}| \cdot 2^{(\omega-1)|U|}|U|^{\mathcal{O}(1)})$ where ω is the matrix multiplication exponent. Since the number of table entries is $\mathcal{O}(2^{7k}n^{13})$, U is at most $7k + 13$ and $\mathcal{A}$ is at most $2^{|U|}$, we have the total time bounded to be $\mathcal{O}(2^{(\omega-1)7k}(7k)^{\mathcal{O}(1)}2^{7k}n^{13}) = \mathcal{O}(2^{\omega 7k}(7k)^{\mathcal{O}(1)}n^{13})$. □

4.1 SETH Lower Bounds

A graph G is called a cluster graph if it is a disjoint union of complete graphs. It can be seen that all cluster graphs are chordal. We define a problem called VERTEX COVER BY CLSVD.

VERTEX COVER BY CLSVD	**Parameter:** k
Input: A graph $G = (V, E)$, $k, \ell \in \mathbb{N}$ and a set $S \subseteq V(G)$ with $\lvert S\rvert \leq k$ such that $G[V \setminus S]$ is a cluster graph.	
Question: Is there a vertex cover of size ℓ in G?	

Assuming SETH, we show that VERTEX COVER BY CLSVD, FVS BY CVD and OCT BY CVD cannot have an $(2-\epsilon)^k n^{\mathcal{O}(1)}$ FPT algorithm. As the class of all cluster graphs is a subclass of the class of chordal graphs, deletion distance to a chordal graph is a smaller parameter. Hence the lower bound also holds for VERTEX COVER BY CVD.

To show the following theorem, we give a parameterized reduction from HITTING SET parameterized by the size of the universe n to VERTEX COVER BY CLSVD and use the fact that assuming SETH, HITTING SET cannot be solved in $(2-\epsilon)^n (n+m)^{\mathcal{O}(1)}$ time for n elements and m sets.

Theorem 5 VERTEX COVER BY CLSVD *cannot be solved in* $(2-\epsilon)^k n^{\mathcal{O}(1)}$ *time for any* $\epsilon > 0$ *assuming SETH.*

Proof We give a reduction from HITTING SET defined as follows.

HITTING SET : In any instance of HITTING SET, we are given a set of elements U with $\lvert U\rvert = n$, a family of subsets $\mathcal{F} = \{F \subseteq U\}$ with $m = \lvert\mathcal{F}\rvert$ and a natural number k. The objective is to find a set $F \subseteq U$, $\lvert F\rvert \leq k$ such that $S \cap F \neq \emptyset$ for all $S \in \mathcal{F}$.

The problem cannot be solved in $(2-\epsilon)^n (n+m)^{\mathcal{O}(1)}$ time assuming SETH [11].

Consider a HITTING SET instance $(U, \mathcal{F})$. We construct an instance of VERTEX COVER BY CLSVD as follows. For each element $u \in U$, we add a vertex v_u. For each set $S \in \mathcal{F}$, we add $\lvert S\rvert$ vertices corresponding to the elements in S. We also make the vertices of S into a clique. Finally, for each element $u \in U$, we add edges from v_u to the vertex corresponding to u for each set in S that contains u. See Fig. 1.

Note that the set of vertices $\bigcup_{u \in U} v_u$ forms a cluster vertex deletion set of size n for the graph G we constructed.

We claim that there is a hitting set of size k in the instance $(U, \mathcal{F})$ if and only if there is a vertex cover of size $k + \sum_{S \in \mathcal{F}} (\lvert S\rvert - 1)$ in G.

Let $X \subseteq U$ be the hitting set of size k. For each set $S \in \mathcal{F}$, mark an element of X which intersects S. Now we create a subset of vertices Y in G consisting of vertices corresponding to elements in X plus the vertices corresponding to all the unmarked elements in S for every set $S \in \mathcal{F}$. Clearly $\lvert Y\rvert = k + \sum_{S \in \mathcal{F}} (\lvert S\rvert - 1)$. We claim that Y is a vertex cover of G. Let us look at an edge of G between an element vertex u and its corresponding copy vertex in S containing u. If u is unmarked in S, then it is covered as the vertex corresponding to u in S is present in Y. If it is marked, then the element v_u is present in Y which covers the edge. All the other edges of G have both endpoints in a set $S \in \mathcal{F}$. Since one of them is unmarked, it belongs to Y which covers the edge.

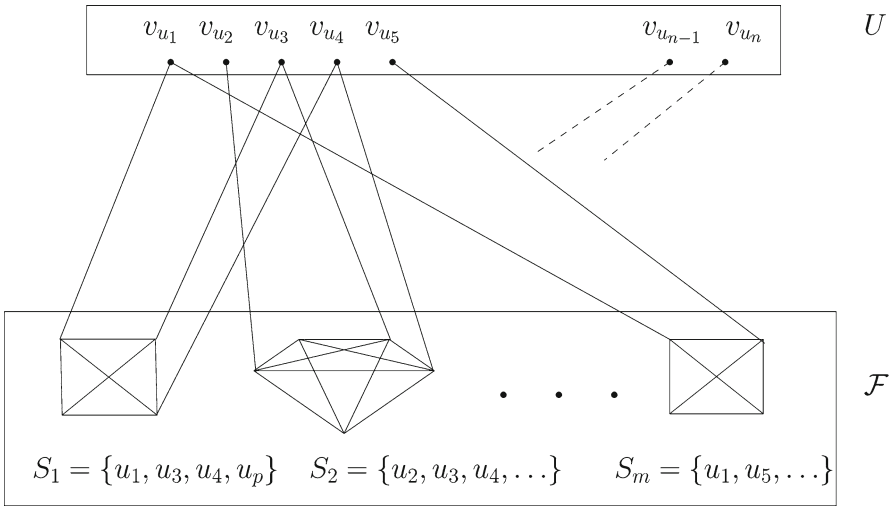

Fig. 1 Reduction from HITTING SET to VERTEX COVER BY CLSVD

Conversely, let Z be a vertex cover of G of size $k + \sum_{S \in \mathcal{F}}(|S| - 1)$. Since the graph induced on vertices of set S forms a clique for each $S \in \mathcal{F}$, Z should contain all the vertices of the clique except one to cover all the edges of the clique. Let us mark these vertices. This means that at least $\sum_{S \in \mathcal{F}}(|S| - 1)$ of the vertices of Z are not element vertices v_u. Now the remaining k vertices of Z should hit all the remaining edges in G. Suppose it contains another vertex x corresponding to an element u in set $S \in \mathcal{F}$. Since x can only cover the edge from x to the element vertex v_u out of the remaining edges, we could remove x and add v_u as it is not present in Z and still get a vertex cover of G of the same size. Hence we can assume, without loss of generality that all the remaining vertices of Z are element vertices v_u. Let X' be the union of the k elements corresponding to these element vertices. We claim that X' is a hitting set of $(U, \mathcal{F})$ of size k. Suppose X' does not hit a set $S \in \mathcal{F}$. Look at the unmarked vertex x in the vertices of S. There is an edge from x to its element vertex v_u. Since $u \notin X'$, this edge is uncovered in G giving a contradiction.

Hence given a HITTING SET instance $(U, \mathcal{F})$, we can construct an instance for VERTEX COVER BY CLSVD with parameter n. Hence, if we could solve VERTEX COVER BY CLSVD in $(2-\epsilon)^k n^{\mathcal{O}(1)}$ time, we can solve HITTING SET in $(2-\epsilon)^n (n + m)^{\mathcal{O}(1)}$ time contradicting SETH. □

The proof of the following theorem works by modifying the reduction in the above proof to replace edges with triangles.

Theorem 6 FVS BY CVD *and* OCT BY CVD *given the modulator cannot be solved in* $(2 - \epsilon)^k n^{\mathcal{O}(1)}$ *time for any* $\epsilon > 0$ *assuming SETH.*

Proof To prove the above theorem, we again give a reduction very similar to the reduction given in the proof of Theorem 5. Consider a HITTING SET instance $(U, \mathcal{F})$. To create an instance of FEEDBACK VERTEX SET BY CVD or ODD CYCLE TRANSVERSAL BY CVD, we replace each edge in the above reduction by a triangle. It can be

easily shown that the graph obtained after removing the vertices corresponding to elements in U forms a chordal graph. The proof follows on similar lines. □

5 Conclusion

Our main contribution is to develop techniques for addressing structural parameterization problems when the modulator is not given. The problem posed by Fellows et al. about whether there is an FPT algorithm for VERTEX COVER parameterized by perfect deletion set with only a promise on the size of the deletion set, is open. Regarding problems parameterized by chordal deletion set size, though our algorithms are based on treewidth DP, we remark that not all problems that have FPT algorithms when parameterized by treewidth necessarily admit an FPT algorithm parameterized by CVD. For example, DOMINATING SET parameterized by treewidth admits an FPT algorithm [12] while DOMINATING SET parameterized by CVD is para-NP-hard as the problem is NP-hard in chordal graphs [5]. Generalizing our algorithms for other problems, for example, for the optimization problems considered by Liedloff et al. [33] would be an interesting direction.

Finally, we believe that this whole notion of permissive problems needs to be explored in many facets of structural parameterizations where finding the modulator is more expensive than solving the problem when the modulator is given.

References

1. Agrawal, A., Lokshtanov, D., Misra, P., Saurabh, S., Zehavi, M.: Polylogarithmic approximation algorithms for weighted-f-deletion problems. ACM Trans. Algorithms **16**(4), 51:1-51:38 (2020)
2. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
3. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015)
4. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: A c^{kn} 5-approximation algorithm for treewidth. SIAM J. Comput. **45**(2), 317–378 (2016)
5. Booth, K.S., Johnson, J.H.: Dominating sets in chordal graphs. SIAM J. Comput. **11**, 191–199 (1982)
6. Brandstädt, A.: On robust algorithms for the maximum weight stable set problem. In: International Symposium on Fundamentals of Computation Theory, pp. 445–458. Springer (2001)
7. Breu, H., Kirkpatrick, D.G.: Unit disk graph recognition is NP-hard. Comput. Geom. **9**(1–2), 3–24 (1998)
8. Cao, Y., Marx, D.: Chordal editing is fixed-parameter tractable. Algorithmica **75**(1), 118–137 (2016)
9. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Math. **86**(1–3), 165–177 (1990)
10. Corneil, D.G., Fonlupt, J.: The complexity of generalized clique covering. Discrete Appl. Math. **22**(2), 109–118 (1988)
11. Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., Wahlström, M.: On problems as hard as CNF-SAT. ACM Trans. Algorithms **12**(3), 41:1-41:24 (2016)
12. Cygan, M., Fomin, F.V., Kowalik, Ł, Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms, vol. 3. Springer, Berlin (2015)
13. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: On multiway cut parameterized above lower bounds. TOCT **5**(1), 3:1-3:11 (2013)
14. Diestel, R.: Graph Theory. Springer, Berlin (2006)
15. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity, vol. 4. Springer, Berlin (2013)

16. Farber, M.: On diameters and radii of bridged graphs. Discrete Math. **73**(3), 249–260 (1989)
17. Fellows, M., Rosamond, F.: The complexity ecology of parameters: an illustration using bounded max leaf number. In: Conference on Computability in Europe, pp. 268–277. Springer, (2007)
18. Fellows, M.R., Jansen, B.M., Rosamond, F.: Towards fully multivariate algorithmics: parameter ecology and the deconstruction of computational complexity. Eur. J. Comb. **34**(3), 541–566 (2013)
19. Fomin, F.V., Golovach, P.A.: Subexponential parameterized algorithms and kernelization on almost chordal graphs. In: 28th Annual European Symposium on Algorithms, ESA 2020, September 7–9, 2020, Pisa, Italy (Virtual Conference), pp. 49:1–49:17 (2020)
20. Fomin, F.V., Kaski, P., Lokshtanov, D., Panolan, F., Saurabh, S.: Parameterized single-exponential time polynomial space algorithm for steiner tree. SIAM J. Discrete Math. **33**(1), 327–345 (2019)
21. Fomin, F.V., Todinca, I., Villanger, Y.: Large induced subgraphs via triangulations and CMSO. SIAM J. Comput. **44**(1), 54–87 (2015)
22. Gaspers, S., Szeider, S.: Backdoors to satisfaction. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) The Multivariate Algorithmic Revolution and Beyond-Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday, volume 7370 of Lecture Notes in Computer Science, pp. 287–317. Springer (2012)
23. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, vol. 57. Elsevier, Amsterdam (2004)
24. Habib, M., Paul, C.: A simple linear time algorithm for cograph recognition. Discrete Appl. Math. **145**(2), 183–197 (2005)
25. Heggernes, P., van't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. Theor. Comput. Sci. **511**, 172–180 (2013)
26. Iwata, Y., Yamaguchi, Y., Yoshida, Y.: 0/1/all CSPs, Half-integral A-path packing, and linear-time FPT algorithms. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 462–473. IEEE (2018)
27. Jacob, A., Panolan, F., Raman, V., Sahlot, V.: Structural parameterizations with modulator oblivion. In: 15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14–18, 2020, Hong Kong, China (Virtual Conference), pp. 19:1–19:18 (2020)
28. Jansen, B.M.: The Power of Data Reduction: Kernels for Fundamental Graph Problems. PhD thesis, Utrecht University (2013)
29. Jansen, B.M., Bodlaender, H.L.: Vertex cover kernelization revisited. Theory Comput. Syst. **53**(2), 263–299 (2013)
30. Jansen, B.M., Raman, V., Vatshelle, M.: Parameter ecology for feedback vertex set. Tsinghua Sci. Technol. **19**(4), 387–409 (2014)
31. Jansen, B.M.P., Pilipczuk, M.: Approximation and kernelization for chordal vertex deletion. SIAM J. Discrete Math. **32**(3), 2258–2301 (2018)
32. Kim, E.J., Kwon, O.: Erdős-pósa property of chordless cycles and its applications. J. Comb. Theory Ser. B **145**, 65–112 (2020)
33. Liedloff, M., Montealegre, P., Todinca, I.: Beyond classes of graphs with "few" minimal separators: FPT results through potential maximal cliques. Algorithmica **81**(3), 986–1005 (2019)
34. Lokshtanov, D., Narayanaswamy, N., Raman, V., Ramanujan, M., Saurabh, S.: Faster parameterized algorithms using linear programming. ACM Trans. Algorithms (TALG) **11**(2), 15 (2014)
35. Majumdar, D., Raman, V.: Structural parameterizations of undirected feedback vertex set: FPT algorithms and kernelization. Algorithmica **80**(9), 2683–2724 (2018)
36. Majumdar, D., Raman, V., Saurabh, S.: Polynomial kernels for vertex cover parameterized by small degree modulators. Theory Comput. Syst. **62**(8), 1910–1951 (2018)
37. Makino, K., Uno, T.: New algorithms for enumerating all maximal cliques. In: Scandinavian Workshop on Algorithm Theory, pp. 260–272. Springer (2004)
38. Raghavan, V., Spinrad, J.P.: Robust algorithms for restricted domains. J. Algorithms **48**(1), 160–172 (2003)
39. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Comb. Theory Ser. B **63**(1), 65–110 (1995)
40. Spinrad, J.P.: Efficient Graph Representations. American Mathematical Society, Providence (2003)