Revisiting Connected Vertex Cover: FPT Algorithms and Lossy Kernels

R. Krithika, Diptapriyo Majumdar, and Venkatesh Raman

The Institute of Mathematical Sciences, HBNI, Chennai, India.
{*rkrithika/diptapriyom/vraman*}*@imsc.res.in*

Abstract

The CONNECTED VERTEX COVER problem asks for a vertex cover in a graph that induces a connected subgraph. The problem is known to be fixed-parameter tractable (FPT), and is unlikely to have a polynomial sized kernel (under complexity theoretic assumptions) when parameterized by the solution size. In a recent paper, Lokshtanov et al. [STOC 2017], have shown an α-approximate kernel for the problem for every $\alpha > 1$, in the framework of approximate or lossy kernelization. In this work, we exhibit lossy kernels and FPT algorithms for CONNECTED VERTEX COVER for parameters that are more natural and functions of the input, and in some cases, smaller than the solution size.

Our first result is a lossy kernel for CONNECTED VERTEX COVER parameterized by the size of a split deletion set. Let n denote the number of vertices in the input graph. We show that

- CONNECTED VERTEX COVER parameterized by the size k of a split deletion set admits an α-approximate kernel with $\mathcal{O}(k + (2k + \lceil \frac{2\alpha-1}{\alpha-1} \rceil)^{\lceil \frac{2\alpha-1}{\alpha-1} \rceil})$ vertices and an algorithm with $\mathcal{O}(3^k n^{\mathcal{O}(1)})$ running time.
- For the special case when the split deletion set is a clique deletion set, the algorithm runs in $\mathcal{O}(2^k n^{\mathcal{O}(1)})$ time and the lossy kernel has $\mathcal{O}(k + \lceil \frac{2\alpha-1}{\alpha-1} \rceil)$ vertices.

To the best of our knowledge, this (approximate) kernel is one of the few lossy kernels for problems parameterized by a structural parameter (that is not solution size). We extend this lossy kernelization to CONNECTED VERTEX COVER parameterized by an incomparable parameter, and that is the size of a clique cover. We show that

- CONNECTED VERTEX COVER parameterized by the size k of a clique cover is W[1]-hard but admits an α-approximate kernel with $\mathcal{O}(k \lceil \frac{2\alpha-1}{\alpha-1} \rceil)$ vertices for every $\alpha > 1$. This is one of the few problems that are not FPT but admit a lossy kernel.

Finally, we consider the size of a cluster deletion set as parameter. We show that

- CONNECTED VERTEX COVER parameterized by the size k of a cluster deletion set is FPT via an algorithm with running time $\mathcal{O}(4^k n^{\mathcal{O}(1)})$. It also admits an α-approximate kernel with $\mathcal{O}(k^2 + \lceil \frac{2\alpha-1}{\alpha-1} \rceil \cdot \frac{k}{\alpha-1} + \lceil \frac{\alpha}{\alpha-1} \rceil \cdot k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$ vertices for every $\alpha > 1$.
- For the special case when the cluster deletion set is a degree-1 modulator, the FPT algorithm runs in $\mathcal{O}(3^k n^{\mathcal{O}(1)})$ time and the lossy kernel has $\mathcal{O}(k^2 + \frac{k}{\alpha-1} + \lceil \frac{\alpha}{\alpha-1} \rceil \cdot k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$ vertices.

1 Introduction, Motivation and Our Results

A *vertex cover* in a graph is a set of vertices that has at least one endpoint from every edge of the graph. In the VERTEX COVER problem, given a graph G and an integer ℓ, the task is to determine if G has a vertex cover of size at most ℓ. Undoubtedly, VERTEX COVER is one of the most well studied problems in parameterized complexity. In this framework, each problem instance is associated with a non-negative integer called *parameter*. The pair consisting of a decision problem and a parameterization is called a *parameterized problem*. A common parameter is a bound on the size of an optimum solution for the problem instance. A problem is said to be *fixed-parameter tractable* (FPT) with respect to the parameter k if it can be solved in $f(k)n^{\mathcal{O}(1)}$ time for some computable function f, where n is the input size. Such an algorithm is called a *parameterized algorithm* or FPT *algorithm*. For convenience, the running time $f(k)n^{\mathcal{O}(1)}$ where f grows super-polynomially with k is denoted as $\mathcal{O}^*(f(k))$. A *kernelization algorithm* is a polynomial-time algorithm that transforms an arbitrary instance of the problem to an equivalent instance of the same problem whose size is bounded by some computable function g of the parameter of the original instance. The resulting instance is called a *kernel* and if g is a polynomial function, then it is called a polynomial kernel and we say that the problem admits a polynomial kernel. In order to classify parameterized problems as being FPT or not, the W-hierarchy: $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$ is defined. It is believed that the subset relations in this sequence are all strict and a parameterized problem that is hard for some complexity class above FPT in this hierarchy is unlikely to be FPT. A parameterized problem is in XP if it has an algorithm with runtime $f(k)n^{g(k)}$ time for some computable functions f and g and it is in para-NP if it has an FPT non-deterministic algorithm. A para-NP-hard problem is not FPT unless P=NP. The complexity classes FPT and para-NP can be viewed as the parameterized analogues of P and NP.

VERTEX COVER is FPT via an easy $\mathcal{O}^*(2^\ell)$ algorithm and after a long race, the current fastest algorithm runs in $\mathcal{O}^*(1.2738^\ell)$ time [6]. VERTEX COVER also has a kernel with $2\ell - \mathcal{O}(\log \ell)$ vertices and $\mathcal{O}(\ell^2)$ edges [21]. In the CONNECTED VERTEX COVER problem, we seek a *connected vertex cover*, i.e., a vertex cover that induces a connected subgraph. It is easy to observe that VERTEX COVER reduces to CONNECTED VERTEX COVER implying that the latter is at least as hard as the former in general graphs. In fact, CONNECTED VERTEX COVER is NP-hard even on bipartite graphs [13] where VERTEX COVER is solvable in polynomial time (Theorem 2.1.1 [11]). However, CONNECTED VERTEX COVER is polynomial-time solvable on chordal graphs and sub-cubic graphs [13,24] and has an $\mathcal{O}^*(2^{\mathcal{O}(t)})$ algorithm when the input graph has treewidth upper bounded by t [2]. The best known parameterized algorithm for CONNECTED VERTEX COVER takes $\mathcal{O}^*(2^\ell)$ time where ℓ is the size of the connected vertex cover that we are looking for [7]. Further, the problem does not admit a polynomial kernel unless the polynomial hierarchy collapses (specifically, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$) [12].

As the goal in parameterized algorithms is to eventually solve the given instance of a problem, the application of a (classical) kernelization algorithm is typically followed by an exact or approximation algorithm that finds a solution for the reduced instance. However, the current definition of kernels provide no insight into how this solution relates to a solution for the original instance and the basic framework is not amenable to be a precursor to approximation algorithms or heuristics. Recently, Lokshtanov et al. [22] proposed a new framework, referred to as *lossy kernelization*, that is a bit less stringent than the notion of polynomial kernels and combines well with approximation algorithms and heuristics. The key new definition in this framework is that of an α-*approximate kernelization*. The precise definitions are deferred to Section 2. Informally, an α-approximate polynomial kernelization (lossy kernelization) is a polynomial time preprocessing algorithm that takes as input an instance of a parameterized problem and outputs another instance of the same problem whose size is bounded by a polynomial function of the parameter of the original instance. Additionally, for every $c \geq 1$, a c-approximate solution for the reduced instance can be turned into a (αc)-approximate solution for the original instance in polynomial time. The authors of [22] exhibit lossy polynomial kernels for several problems that do not admit classical polynomial kernels including CONNECTED VERTEX COVER parameterized by the solution size. In this paper, we extend this lossy kernel for CONNECTED VERTEX COVER to parameters that are more natural and functions of the input, and in some cases, smaller than the solution size.

In the initial work on parameterized complexity, the parameter was almost always the solution size (with treewidth being a notable exception). A recent trend is to study the complexity of the given problem with respect to structural parameters that are more likely to be small in practice. Also, once a problem is shown to be FPT or to have a polynomial sized kernel by a parameterization, it is natural to ask whether the problem is FPT (and admits a polynomial kernel) when parameterized by a smaller parameter. Similarly, once a problem is shown to be W-hard by a parameterization, it is natural to ask whether the problem is FPT when parameterized by a larger parameter. Structural parameterizations of VERTEX COVER [3, 18], FEEDBACK VERTEX SET [20] and GRAPH COLORING [19] have been explored extensively. We refer to [17] and [18] for a detailed introduction to the whole program. A parameter that has gained significant attention recently is the size of a *modulator* to a family of graphs. Let $\mathcal{F}$ denote a hereditary graph class (which is closed under induced subgraphs) on which VERTEX COVER is polynomial-time solvable. Suppose S is a set of vertices (called an $\mathcal{F}$-modulator) such that $G - S \in \mathcal{F}$. Then, VERTEX COVER is FPT when parameterized by the size of $|S|$. Following is an easy $\mathcal{O}^*(2^{|S|})$ algorithm: guess the intersection of the required solution with the modulator and solve the problem on the remaining graph in polynomial time. In contrast, a similar generic result does not seem easy for CONNECTED VERTEX COVER and a study of such structural parameterizations for CONNECTED VERTEX COVER is another goal of this paper.

Our Results. The starting point of our study is to extend the lossy kernelization known for CONNECTED VERTEX COVER parameterized by solution size to CONNECTED VERTEX COVER parameterized by a smaller parameter. A *split graph* is a graph whose vertex set can be partitioned into a clique and an independent set. Split graphs can be recognized in polynomial time and such a partition can be obtained in the process [16]. A *split deletion set* is a set of vertices whose deletion results in a split graph. CONNECTED VERTEX COVER has an easy polynomial time algorithm in split graphs, and it is easy to verify that the size of minimum connected vertex cover is at least the size of the minimum vertex cover which is at least the size of the minimum split deletion set. We show that

- CONNECTED VERTEX COVER parameterized by the size k of a split deletion set is FPT via an $\mathcal{O}^*(3^k)$ algorithm and that the α-approximate kernel for CONNECTED VERTEX COVER parameterized by solution size can be extended in a non-trivial way to an α-approximate kernel with $\mathcal{O}(k + (2k + \lceil \frac{2\alpha-1}{\alpha-1} \rceil)^{\lceil \frac{2\alpha-1}{\alpha-1} \rceil})$ vertices for every $\alpha > 1$. This adds to the small list of problems for which lossy kernels with respect to structural parameterizations are known.

- For the special case when the split deletion set is a *clique deletion set* (a set of vertices whose deletion results in a complete graph), the algorithm runs in $\mathcal{O}^*(2^k)$ time and the lossy kernel has $\mathcal{O}(k + \lceil \frac{2\alpha-1}{\alpha-1} \rceil)$ vertices (linear size).

Then, we consider CONNECTED VERTEX COVER parameterized by the size of a *clique cover*. A *clique cover* of a graph is a partition of its vertex set such that each part induces a clique. We show that

- CONNECTED VERTEX COVER parameterized by the size k of a clique cover is W[1]-hard but admits an α-approximate kernel with $\mathcal{O}(k\lceil\frac{2\alpha-1}{\alpha-1}\rceil)$ vertices for every $\alpha > 1$. This is one of the few problems that are not FPT but admit a lossy kernel.

Finally, we consider a parameter related (structurally) to clique cover size, namely, the size of a *cluster deletion set* and show that CONNECTED VERTEX COVER is FPT with respect to this parameter. A *cluster graph* is a graph in which every component is a complete graph and a cluster deletion set is a set of vertices whose deletion results in a cluster graph. We show that

- CONNECTED VERTEX COVER parameterized by the size k of a cluster deletion set is FPT via an $\mathcal{O}^*(4^k)$ algorithm and admits an α-approximate kernel with $\mathcal{O}(k^2 + \lceil\frac{2\alpha-1}{\alpha-1}\rceil \cdot \frac{k}{\alpha-1} + \lceil\frac{\alpha}{\alpha-1}\rceil \cdot k^{\lceil\frac{\alpha}{\alpha-1}\rceil})$ vertices for every $\alpha > 1$.
- For the special case when the cluster deletion set is a degree-1 modulator (a subset of vertices whose deletion results in a graph with maximum degree 1), the algorithm runs in $\mathcal{O}^*(3^k)$ time and the lossy kernel has $\mathcal{O}(k^2 + \frac{k}{\alpha-1} + \lceil\frac{\alpha}{\alpha-1}\rceil \cdot k^{\lceil\frac{\alpha}{\alpha-1}\rceil})$ vertices.

A summary of the results and the relation between the parameters are illustrated in Figure 2 in the conclusion.

Techniques. Our FPT algorithms involve reductions to the STEINER TREE problem on bipartite graphs. Given a graph G, an integer p and a set T (called *terminals*) of vertices of G, the STEINER TREE problem is the task of determining whether G contains a tree (called *Steiner tree*) on at most p vertices that contains T. A *bipartite graph* is a graph whose vertex set can be partitioned into 2 independent sets. Such a partition is called a *bipartition.* We use the following result known for STEINER TREE on bipartite graphs.

Lemma 1. [7] *Given a connected bipartite graph G on n vertices, m edges and bipartition (P, Q), there is an algorithm running in $\mathcal{O}(2^{|Q|} \cdot k(m+n))$ time that computes a set $X \subseteq V(G)$ of at most k vertices such that $Q \subseteq X$ and $G[X]$ is connected.*

Our lower bound results are based on the hardness results known for the SET COVER problem. Given a family $\mathcal{F}$ of subsets of a universe U and a positive integer ℓ, the SET COVER problem is the task of determining whether there is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most ℓ such that $\bigcup_{X\in\mathcal{F}'} X = U$. SET COVER can be solved in $\mathcal{O}^*(2^{|U|})$ time by a dynamic programming routine [14] and the *Set Cover Conjecture* states that no $\mathcal{O}^*((2-\epsilon)^{|U|})$ time algorithm exists for any $\epsilon > 0$ [8]. Our lower bound results are based on the following result that is a consequence of this conjecture.

Lemma 2. [8] CONNECTED VERTEX COVER *parameterized by the solution size ℓ has no $O^*((2-\epsilon)^\ell)$ time algorithm under the Set Cover Conjecture.*

Let n and m denote the number of vertices and edges, respectively, in the input graph. For each of the parameterized problems considered in this paper, we assume that the modulator (whose size is the parameter) under consideration is given as part of the input. In most cases, this assumption is reasonable. For instance, there is an algorithm by Cygan and Pilipczuk [10] that runs in $\mathcal{O}(1.2732^k \cdot k^{\mathcal{O}(\log k)} + n^3)$ time and returns a split deletion set of size at most k (or decides that no such set exists). In the case of cluster deletion set size as parameter, we use the algorithm by Boral et al. [5] that runs in $\mathcal{O}(1.9102^k(n + m))$ time and returns a cluster deletion set of size at most k, if one exists. Also, a degree-1 modulator S of size at most k, if one exists, can be obtained in $\mathcal{O}(1.882^k + nm)$ time using the algorithm by Wu [25].

2 Preliminaries

We refer to [11] for graph theoretic terms and notation that are not explicitly defined here. We use $[r]$ to denote the set $\{1, 2, \dots, r\}$. Given a finite set A, we use $\binom{A}{k}$ to denote the collection of subsets of A containing exactly k elements and we use $\binom{A}{\leq k}$ to denote the collection of subsets of A containing at most k elements. We use n to denote the number of vertices and m to denote the number of edges of graph G.

For a graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively. Two vertices u, v are said to be *adjacent* if there is an edge $\{u, v\}$ in the graph. The neighborhood of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v and its degree $d_G(v)$ is $|N_G(v)|$. The subscript in the notation for neighborhood and degree is omitted if the graph under consideration is clear. For a set $S \subseteq V(G)$, $G - S$ denotes the graph obtained by deleting S from G and $G[S]$ denotes the subgraph of G induced by set S. The *contraction* operation of an edge $e = \{u, v\}$ in G results in the deletion of u and v and the addition of a new vertex w adjacent to vertices that were adjacent to either u or v. Any parallel edges added in the process are deleted so that the graph remains simple. This operation is extended to a subset of edges and the resultant graph is oblivious to the contraction sequence. A *path* is a sequence of distinct vertices where every consecutive pair of vertices are adjacent. A set of pairwise non-adjacent vertices is called as an *independent set* and a set of pairwise adjacent vertices is called as a *clique*. A *complete graph* is a graph whose vertex set is a clique. The number of components of G is denoted by $\#\text{comp}(G)$ and by convention if G is connected then $\#\text{comp}(G)$ is 1. Two non-adjacent vertices u and v are called *false twins* if $N(u) = N(v)$.

Lossy Kernelization: Parameterized complexity terminology and definitions not stated here can be found in [9]. We now state terminology and definitions related to lossy kernelization given in [22]. A *parameterized minimization problem* is a computable function $\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \mapsto \mathbb{R} \cup \{\pm\infty\}$. The instances of Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$ and a solution for (I, k) is a string $S \in \Sigma^*$ such that $|S| \leq |I| + k$. The *value* of a solution S is $\Pi(I, k, S)$. The *optimum value* of (I, k) is $\text{OPT}_\Pi(I, k) = \min_{S \in \Sigma^*, |S| \leq |I|+k} \Pi(I, k, S)$, and an *optimum solution* for (I, k) is a solution S such that $\Pi(I, k, S) = \text{OPT}_\Pi(I, k)$. A *parameterized maximization problem* is defined in a similar way. We will omit the subscript Π in the notation for optimum value if the problem under consideration is clear from context. For each of the problems considered in this paper, we define the corresponding parameterized minimization problem as follows.

$$\text{CVC}((G, S), k, T) = \begin{cases} -\infty & \text{if } |S| > k \text{ or } S \text{ is not an } \mathcal{F}\text{-modulator} \\ \infty & \text{if } T \text{ is not a connected vertex cover} \\ |T| & \text{otherwise} \end{cases}$$

The cases specified in the definition are ordered, i.e., given (G, S), an integer k and a set $T \subseteq V(G)$, $\text{CVC}((G, S), k, T)$ is assigned the first value applicable. A *strict α-approximate polynomial-time preprocessing algorithm* for Π is a pair of polynomial-time algorithms, called the *reduction algorithm* and the *solution lifting algorithm*, that satisfy the following properties.

- Given an instance (I, k) of Π, the reduction algorithm computes an instance (I', k') of Π.
- Given the instances (I, k) and (I', k') of Π, and a solution S' to (I', k'), the solution lifting algorithm computes a solution S to (I, k) such that $\frac{\Pi(I,k,S)}{\text{OPT}(I,k)} \leq \max\{\frac{\Pi(I',k',S')}{\text{OPT}(I',k')}, \alpha\}$.

A *reduction rule* is the execution of the reduction algorithm on an instance, and we say that it is applicable on an instance if the output instance is different from the input instance. An

α-approximate kernelization (or α-approximate kernel) for Π is an α-approximate polynomial-time preprocessing algorithm such that the size of the output instance is upper bounded by a computable function $g : \mathbb{N} \to \mathbb{N}$ of k. A reduction rule is said to be *α-safe* for Π if there is a solution lifting algorithm, such that the rule together with this algorithm constitute a strict α-approximate polynomial-time preprocessing algorithm for Π. A reduction rule is *safe* if it is 1-safe. Observe that this definition is more strict that the definition of safeness in classical kernelization. A *polynomial-size approximate kernelization scheme (PSAKS)* for Π is a family of α-approximate polynomial kernelization algorithms for each $\alpha > 1$. Note that, the size of an output instance of a PSAKS, when run on (I, k) with approximation parameter α, must be upper bounded by $f(\alpha) \cdot k^{g(\alpha)}$ for some functions f and g independent of $|I|$ and k. A PSAKS is said to be *time efficient* if the reduction algorithm and the solution lifting algorithm run in $f(\alpha) \cdot |I|^c$, for some computable function f. We encourage the reader to see [22] for a more comprehensive discussion of these ideas and definitions. We end the preliminaries section by stating the following result known from [22].

Lemma 3 (*Theorem 1 of [22]*). *For every $\alpha \geq 1$ and decidable parameterized optimization problem Π, Π admits a polynomial-time α-approximation algorithm if and only if Π has an α-approximate kernel of constant size.*

As a consequence of this result and the fact that Connected Vertex Cover has a 2-approximation algorithm, we focus on designing α-approximate kernels for $1 < \alpha < 2$. For $\alpha \geq 2$, Lemma 3 gives the required α-approximate kernel.

3 Connected Vertex Cover parameterized by Split Deletion Set

In this section, we describe an FPT algorithm and a lossy polynomial kernel for Connected Vertex Cover parameterized by the size of a split deletion set. Recall that a split deletion set is a set of vertices whose deletion results in a split graph. Throughout this section (and subsequent sections), we use $|S| \leq k$. So, we will use $|S|$ and k interchangeably to denote the same thing.

3.1 An FPT algorithm

In this subsection we describe an $\mathcal{O}^*(3^k)$ time algorithm for Connected Vertex Cover parameterized by the size k of a split deletion set.

Theorem 4. *Given a graph G on n vertices and m edges, a split deletion set S and a positive integer ℓ, there is an algorithm that determines whether G has a connected vertex cover of size at most ℓ in $\mathcal{O}(3^{|S|} \cdot (|S|^2 + \ell)(n + m))$ time. Moreover, no $O^*((2 - \epsilon)^{|S|})$ time algorithm exists for any $\epsilon > 0$ for this problem under the Set Cover Conjecture. Further, if S is a clique deletion set, then there is an algorithm that runs in $\mathcal{O}(2^{|S|} \cdot |S|^2(n + m))$ time.*

Proof. Let H denotes the split graph $G - S$ whose vertex set can be partitioned into a clique C and an independent set I. This partition of H into C and I can be computed in $O(n - k + m)$ time. Let $|S| = k$ and let X^* be a connected vertex cover of G of size at most ℓ (if one exists). We show that once we guess the intersection of X^* with S and C, the problem reduces to a version of the Steiner tree problem with at most $|S|$ terminals. As X^* can afford to exclude at most one vertex from C, there are at most $|C| + 1$ choices for $X^* \cap C$. Also, there are at most 2^k choices for $X^* \cap S$. Consider one such choice (Y, Z) where $Y = X^* \cap C$ and $Z = X^* \cap S$. We will extend $T = Y \cup Z$ into a connected vertex cover of G.

If $(C \setminus Y) \cup (S \setminus Z)$ is not an independent set, no such connected vertex cover exists and we skip to the next choice of (Y, Z). We can check if $(C \setminus Y) \cup (S \setminus Z)$ is an independent set or not in $\mathcal{O}(|S|^2)$ time as there are at most $|S| + 1$ vertices in $(S - X^*) \cup (C - X^*)$.

Let us first consider the case when $I = \emptyset$. That is, S is a clique deletion set of G. In this case, as we have already made our choice in S and C, and so if $G[T]$ is not connected, we can skip to the next choice of (Y, Z). If none of the choices leads to the required solution, we declare that G has no connected vertex cover of size at most ℓ. The overall running time of the algorithm is $\mathcal{O}(2^k \cdot k^2(n + m))$.

When $I \neq \emptyset$, observe that $G[T]$ has at most $|Z| + 1$ components. Let R denote the set $(N(C \setminus Y) \cup N(S \setminus Z)) \setminus (Y \cup Z)$. Clearly, $R \subseteq I$ and has to be included in the solution (see Figure 1). If there is a vertex $r \in R$ that has no neighbors in T, then r cannot be connected to T by vertices from I as $R \subseteq I$. Therefore, if there is such a vertex r, then we skip to the next choice of (Y, Z). Otherwise, by adding R to T, $\#\text{comp}(G[T])$ cannot increase in this process. The problem now reduces to finding a set $J \subseteq I \setminus R$ such that $G[T \cup R \cup J]$ is connected.

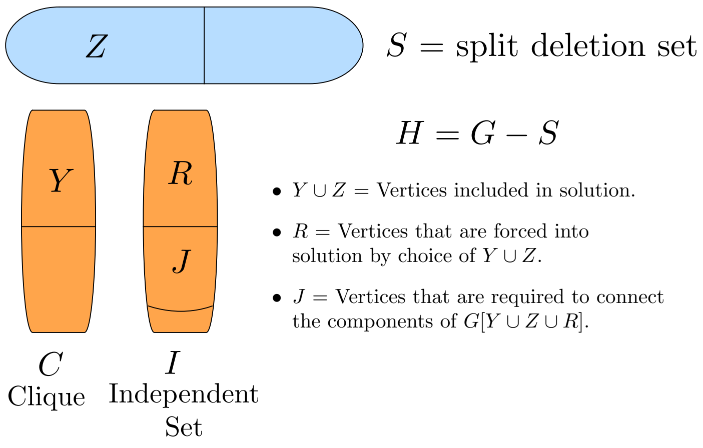
Z
S = split deletion set
$H = G - S$
Y
R
J
C
Clique
I
Independent Set
• $Y \cup Z$ = Vertices included in solution.
• R = Vertices that are forced into solution by choice of $Y \cup Z$.
• J = Vertices that are required to connect the components of $G[Y \cup Z \cup R]$.

Figure 1: Connected vertex cover parameterized by split deletion set.

Now consider the bipartite graph $\widehat{G}$ with bipartition $(T^*, I \setminus R)$ where each vertex of T^* corresponds to a component of $G[T \cup R]$. We draw an edge between a vertex $p \in I \setminus R$ and a vertex c in T^* if there is an edge in G between p and a vertex in the component corresponding to c. The task now is to find a minimum Steiner tree of $\widehat{G}$ with terminal set T^* which can be done in $\mathcal{O}(2^{|T^*|}\ell(n + m))$ time using Lemma 1. Also, whether every vertex of R has some neighbor in T or not can be checked in $\mathcal{O}(n + m)$ time. Therefore, the overall running time of the algorithm is $\sum_{i=0}^{k} \binom{k}{i} \cdot (k^2 + 2^i \ell(n + m))$ where the first term in the product is the number of choices for $X^* \cap S$ where $|X^* \cap S| = i$ and the second term is the time taken to check if $(C \setminus Y) \cup (S \setminus Z)$ is an independent set or not and to find a minimum Steiner tree for an instance with at most i terminals. Thus, the algorithm runs in $O(3^{|S|} \cdot (|S|^2 + \ell)(n + m))$ time.

As an edgeless graph is a split graph, we have that the size of a minimum connected vertex cover is at least the size of a minimum connected split deletion set. Therefore, the claimed lower bound follows from Lemma 2. $\square$

In order to obtain a tighter and refined analysis of the running time of the above algorithm, we generalize the following lemma. Here $\mathcal{VC}(H)$ denotes the set of vertex covers of H.

Lemma 5. [7] *Let* H *be a connected graph on* h *vertices. Then,* $\sum\limits_{C \in \mathcal{VC}(H)} 2^{\#\text{comp}(H[C])} \leq 3 \cdot 2^{h-1}$.

We next generalize Lemma 5 to graphs that are not necessarily connected.

Lemma 6. *Let* H *be a graph on* h *vertices with* $\#\text{comp}(H) = d$*. Then,* $\sum\limits_{C \in \mathcal{VC}(H)} 2^{\#\text{comp}(H[C])} \leq 3^d 2^{h-d}$.

Proof. Let $H_1, \cdots, H_d$ be the distinct components of H and let h_i denote the number of vertices in H_i. Any vertex cover of H is the union of the vertex covers of H_i, $1 \leq i \leq d$. Consider a vertex cover C of H. That is, $C \in \mathcal{VC}(H)$. Let C_i be the set of vertices of C that are in H_i. Observe that C_i is a vertex cover of H_i for each $i \in [d]$. Then, $\#\text{comp}(H[C]) = \sum\limits_{i=1}^{d} \#\text{comp}(H_i[C_i])$. Thus, $2^{\#\text{comp}(H[C])} = 2^{\sum_{i=1}^{d} \#\text{comp}(H_i[C_i])} = \prod\limits_{i=1}^{d} 2^{\#\text{comp}(H_i[C_i])}$. As the number of vertex covers of H is equal to the product of the numbers of vertex covers of H_i, $i \in [d]$, we have $|\mathcal{VC}(H)| = \prod\limits_{i=1}^{d} |\mathcal{VC}(H_i)|$. So, we have that $\sum\limits_{C \in \mathcal{VC}(H)} 2^{\#\text{comp}(H[C])} = \prod\limits_{i=1}^{d} \sum\limits_{C \in \mathcal{VC}(H_i)} 2^{\#\text{comp}(H_i[C])}$. As each H_i is connected, the following bound follows from Lemma 5.

$$\sum_{C \in \mathcal{VC}(H)} 2^{\#\text{comp}(H[C])} = \prod_{i=1}^{d} \sum_{C \in \mathcal{VC}(H_i)} 2^{\#\text{comp}(H_i[C])} \leq \prod_{i=1}^{d} 3 \cdot 2^{h_i - 1} = 3^d \cdot 2^{h-d}$$

□

Using Theorem 4 and Lemma 6, we have the following result.

Corollary 7. *Given a graph* G*, a split deletion set* S *with* $\#\text{comp}(G[S]) = d$ *and a positive integer* ℓ*, there is an algorithm that determines whether* G *has a connected vertex cover of size at most* ℓ *in* $O(3^d \cdot 2^{|S|-d} \cdot (|S|^2 + \ell)(n+m))$ *time.*

Proof. The running time of the algorithm presented in Theorem 4 is upper bounded (ignoring polynomial factors) by $\sum\limits_{Z \in \mathcal{VC}(G[S])} 2^{\#\text{comp}(G[Z])}(|S|^2 + \ell)(n+m)$ which in turn is upper bounded by $3^d \cdot 2^{|S|-d} \cdot (|S|^2 + \ell)(n+m)$ from Lemma 6. Therefore, the claimed bound follows. □

Using Lemma 2, Corollary 7 and the fact that a vertex cover of size ℓ (if one exists) can be obtained in $\mathcal{O}(1.2738^\ell + \ell n)$ time [6], we have the following result.

Corollary 8. *Given a graph* G*, a vertex cover* S *with* $\#\text{comp}(G[S]) = d$ *and a positive integer* ℓ*, there is an algorithm that determines whether* G *has a connected vertex cover of size at most* ℓ *in* $O^*(3^d 2^{|S|-d})$ *time. Further, no* $O^*((2-\epsilon)^{|S|})$ *time algorithm exists for this problem for any* $\epsilon > 0$ *under the Set Cover Conjecture.*

3.2 A Lossy Kernel

It is known that Vertex Cover parameterized by the size of a clique deletion set has no polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP}/\text{poly}$ [4]. As Vertex Cover reduces to Connected Vertex Cover by just adding a universal vertex to the graph, we have the following observation.

Observation 1. Connected Vertex Cover *parameterized by the size of a clique deletion set has no polynomial kernel unless* $\mathsf{NP} \subseteq \mathsf{coNP}$*/poly.*

In this section, we show that Connected Vertex Cover parameterized by the size of a split deletion set admits a lossy polynomial kernel. As a consequence, we show that Connected Vertex Cover parameterized by the size of a clique deletion set admits a lossy linear kernel. Following is the definition of the corresponding parameterized minimization problem.

$$\mathsf{CVC}((G,S),k,T) = \begin{cases} -\infty & \text{if } |S| > k \text{ or } G - S \text{ is not a split graph} \\ \infty & \text{if } T \text{ is not a connected vertex cover} \\ |T| & \text{otherwise} \end{cases}$$

Without loss of generality, we assume that G has no isolated vertices as no minimum connected vertex cover contains them. Given $\alpha > 1$, let d be the minimum integer greater than 1 such that $\alpha \geq \frac{d-1}{d-2}$. That is, $d = \lceil \frac{2\alpha-1}{\alpha-1} \rceil$. Let H denote the split graph $G - S$ whose vertex set can be partitioned into a clique C and an independent set I. First, we bound the number of vertices in C by applying Reduction Rule 3.1. Recall that we only consider values of α that are between 1 and 2. For $\alpha \geq 2$, Lemma 3 gives the required α-approximate kernel. As $1 < \alpha < 2$, we have $d = \lceil \frac{2\alpha-1}{\alpha-1} \rceil \geq 3$.

Reduction Rule 3.1. *If* $|C| \geq d$*, then add a new vertex* u_C *adjacent to* $N(C)$ *and delete* C *to get the graph* G'*. The resulting instance is* (G', S).

Observe that S continues to be a split deletion set of G' as a result of applying this rule since $V(G' - S)$ can be partitioned into the clique $\{u_C\}$ and the independent set I.

Lemma 9. *Reduction Rule 3.1 is* α*-safe and can be applied in polynomial time.*

Proof. Consider a solution D' of the reduced instance. If $u_C \in D'$, then the solution lifting algorithm returns $D = (D' \setminus \{u_C\}) \cup C$ which is a connected vertex cover of G such that $\mathsf{CVC}((G,S),k,D) = \mathsf{CVC}((G',S),k,D') - 1 + |C|$. Otherwise, we know that $N(C) \subseteq D'$. Let x be a vertex in C such that $C \setminus \{x\}$ has a neighbor in $V(G') \setminus C$. Note that such a vertex exists in C since $d \geq 3$ and $|C| \geq d$. Then, $D = D' \cup (C \setminus \{x\})$ is a connected vertex cover of G such that $\mathsf{CVC}((G,S),k,D) = \mathsf{CVC}((G',S),k,D') + |C| - 1$. In any case, we have shown that $\mathsf{CVC}((G,S),k,D) = \mathsf{CVC}((G',S),k,D') + |C| - 1$. Now, consider an optimum solution D^* for the original instance. Clearly, $|D^* \cap C| \geq |C| - 1$. Then, $(D^* \setminus C) \cup \{u_C\}$ is a connected vertex cover of G'. Hence, $\mathrm{OPT}((G',S),k) \leq \mathrm{OPT}((G,S),k) - |C| + 1 + 1$. Thus, $\frac{\mathsf{CVC}((G,S),k,D)}{\mathrm{OPT}((G,S),k)} \leq \frac{\mathsf{CVC}((G',S),k,D')+|C|-1}{\mathrm{OPT}((G',S),k)+|C|-2}$. As $|C| \geq 3$, we have $|C| - 1, |C| - 2 > 0$ and $\frac{|C|-1}{|C|-2} \leq \frac{d-1}{d-2} \leq \alpha$. Therefore, combining these bounds, we have the following.

$$\frac{\mathsf{CVC}((G,S),k,D)}{\mathrm{OPT}((G,S),k)} \leq \max\left\{\frac{\mathsf{CVC}((G',S),k,D')}{\mathrm{OPT}((G',S),k)}, \frac{|C|-1}{|C|-2}\right\} \leq \max\left\{\frac{\mathsf{CVC}((G',S),k,D')}{\mathrm{OPT}((G',S),k)}, \alpha\right\}$$

Hence, this reduction rule is α-safe. Further, this rule can be applied in polynomial time as this application involves standard graph theoretic operations like determining the size of C and deleting C from the graph. Therefore, the reduction algorithm can be implemented in polynomial time. Finally, the solution lifting algorithm modifies the given solution using at most one deletion and at most $|C|$ additions. Hence, it can also be implemented in polynomial time. This completes the proof. $\square$

Observe that the application of Reduction Rule 3.1 does not make any vertex isolated in the reduced graph. Further, when S is a clique deletion set and this rule is not applicable, it follows that $G - S$ has at most $d - 1$ vertices. This leads to the following result.

Theorem 10. Connected Vertex Cover *parameterized by the size* k *of a clique deletion set has a time efficient PSAKS with at most* $k + \lceil \frac{2\alpha-1}{\alpha-1} \rceil$ *vertices.*

Proof. Let (G, S) be an instance on which Reduction Rule 3.1 is not applicable. Then, $|S| \leq k$ and $|V(G - S)| \leq d$ where $d = \lceil \frac{2\alpha-1}{\alpha-1} \rceil$. Therefore, $|V(G)| \leq k + d$. $\square$

Let us now consider the case when S is not a clique deletion set.

Lemma 11. *Let* (G, S) *be an instance on which Reduction Rule 3.1 is not applicable. Then,* G *has a connected vertex cover of size at most* $2k + d - 1$*.*

Proof. If Reduction Rule 3.1 is not applicable, then $|C| \leq d - 1$. Now, $T = S \cup C$ is a vertex cover of G such that $G[T]$ has at most $|S| + 1$ components. If $G[T]$ is not connected, then we can add at most $|S|$ vertices from $V(G) \setminus T$ to T so that $G[T]$ becomes connected. Such a choice of vertices exists as G is connected. Hence, it follows that G has a connected vertex cover of size at most $|T| + |S| \leq 2k + d - 1$. $\square$

Now, consider an instance (G, S) on which Reduction Rule 3.1 is not applicable. From Lemma 11, we have $\mathrm{OPT}((G, S), k) \leq 2k + d - 1$. Therefore, any vertex with at least $2k + d$ neighbors is present in any optimal solution. We define a partition of vertices of G into the following three parts.

- $B = \{u \in V(G) \mid d(u) \geq 2k + d\}$
- $I_B = \{v \in V(G) \setminus B \mid N(v) \subseteq B\}$
- $R = V(G) \setminus (B \cup I_B)$

We apply the following two reduction rules which are known from [22] to bound I_B (which is an independent set).

Reduction Rule 3.2. *If there exists* $u \in I_B \cap V(G - S)$ *such that* $d_G(u) \geq d$*, then delete* $N_G[u]$ *and add a new vertex* w *adjacent to every vertex in* $N_G(N_G(u)) \setminus \{u\}$*. Further, add* $2k + d$ *new vertices* W *adjacent to* w *to get the graph* G'*. The resulting instance is* $(G', (S \cup \{w\}) \setminus N(u))$*.*

Observe that $S' = (S \cup \{w\}) \setminus N(u)$ is a split deletion set of G' as $V(G' - S')$ can be partitioned into the clique $C \cap V(G' - S')$ and independent set $(I \cap V(G' - S')) \cup W$. Further, as $|C| \leq d - 1$ and $d_G(u) \geq d$, it follows that $N(u) \cap S \neq \emptyset$. As we add at most one vertex to S and delete at least one vertex from S to get S', we have $|S'| \leq |S|$.

Lemma 12. *Reduction Rule 3.2 is* α*-safe and can be applied in polynomial time.*

Proof. Consider a solution D' of the reduced instance. If $w \notin D'$, the solution lifting algorithm returns a connected vertex cover D of G of size at most $2k + d - 1$ obtained from Lemma 11. Then, we have $\mathrm{CVC}((G', S'), k, D') \geq 2k + d$ since w has at least $2k + d$ neighbors and $\mathrm{CVC}((G, S), k, D) \leq 2k + d - 1$ implying $\mathrm{CVC}((G, S), k, D) \leq \mathrm{CVC}((G', S'), k, D')$. Otherwise, we have $w \in D'$ and the solution lifting algorithm returns $D = (D' \setminus (W \cup \{w\})) \cup N[u]$ which is a connected vertex cover of G such that $\mathrm{CVC}((G, S), k, D) \leq \mathrm{CVC}((G', S'), k, D') - 1 + |N[u]|$. Next, consider an optimum solution D^* for the original instance. Clearly, $|D^*| \leq 2k + d - 1$ from Lemma 11. Thus, $B \subseteq D^*$. In particular, $N_G(u) \subseteq D^*$. Then, $(D^* \setminus N_G[u]) \cup \{w\}$ is a connected vertex cover of G'. Hence, $\mathrm{OPT}((G', S'), k) \leq \mathrm{OPT}((G, S), k) - |N_G(u)| + 1$. As $|N(u)| \geq d$, we have that $\frac{|N(u)|}{|N(u)|-1} \leq \frac{d}{d-1} \leq \frac{d-1}{d-2} \leq \alpha$. By an argument similar to that of Lemma 9, we have that $d \geq 3$. Thus, $|N(u)|, |N(u)| - 1 > 0$. Combining the above mentioned bounds, we have $\frac{\text{CVC}((G,S),k,D)}{\text{OPT}((G,S),k)} \leq \frac{\text{CVC}((G',S'),k,D')+|N(u)|}{\text{OPT}((G',S'),k)+|N(u)|-1} \leq \max\left\{\frac{\text{CVC}((G',S'),k,D')}{\text{OPT}((G',S'),k)}, \frac{|N(u)|}{|N(u)|-1}\right\} \leq \max\left\{\frac{\text{CVC}((G',S'),k,D')}{\text{OPT}((G',S'),k)}, \alpha\right\}$ as $|N(u)| \geq d$. Therefore, this reduction rule is α-safe.

To apply this rule, we need to check if there is a vertex $u \in I_B \cap V(G-S)$ such that $d_G(u) \geq d$. This can be done in polynomial time by examining all vertices of $I_B \cap V(G-S)$. Once such a vertex is found, we delete $N_G[u]$ from G, add a new vertex w adjacent to all vertices in $N_G(N_G(u)) \setminus \{u\}$ and add a set of new $2k+d$ pendant vertices adjacent to w. Thus, this rule can be implemented in polynomial time. Further, as the solution lifting algorithm consists of standard graph operations, it can also be applied in polynomial time. This completes the proof. $\square$

Recall that two non-adjacent vertices u and v are called false twins if $N(u) = N(v)$.

Reduction Rule 3.3. *If there exists* $x \in I_B \cap V(G-S)$ *such that* x *has at least* $2k+d$ *false twins in* $I_B \cap V(G-S)$*, then delete* x*. The resulting instance is* $(G - \{x\}, S \setminus \{x\})$.

Observe that $S' = S \setminus \{x\}$ continues to be a split deletion set of $G' = G - \{x\}$ as a result of applying this rule.

Lemma 13. *Reduction Rule 3.3 is* 1*-safe and can be applied in polynomial time.*

Proof. Consider a solution D' of the reduced instance. If $|D'| \geq 2k+d$, the solution lifting algorithm returns a connected vertex cover D of G of size at most $2k+d-1$ obtained from Lemma 11. Then, we have $\text{CVC}((G',S'),k,D') \geq 2k+d$ and $\text{CVC}((G,S),k,D) \leq 2k+d-1$ implying $\text{CVC}((G,S),k,D) \leq \text{CVC}((G',S'),k,D')$. Otherwise, it follows that one of the false twins of x, say y, is excluded from D'. Thus, $N(y) \subseteq D'$ implying that $N(x) \subseteq D'$. Then, the solution lifting algorithm returns $D = D'$ which is a connected vertex cover of G such that $\text{CVC}((G,S),k,D) = \text{CVC}((G',S'),k,D')$. Next, consider an optimum solution D^* for the original instance. Clearly, $|D^*| \leq 2k+d-1$ from Lemma 11. Thus, either x and one of its false twins y is excluded from D^* or two of the false twins of x, say y and z are excluded from D^*. In any case, we have another optimal connected vertex cover D^{**} of G' that excludes x. Hence, $\text{OPT}((G',S'),k) \leq \text{OPT}((G,S),k)$. Combining these bounds, we have $\frac{\text{CVC}((G,S),k,D)}{\text{OPT}((G,S),k)} \leq \frac{\text{CVC}((G',S'),k,D')}{\text{OPT}((G',S'),k)}$. So, this reduction rule is 1-safe.

As the reduction algorithm and the solution lifting algorithm involve standard graph operations (like computing $N_G(x)$ for each $x \in I_B \cap V(G-S)$) that can be implemented in polynomial time, this rule can be applied in polynomial time. This completes the proof. $\square$

Now, we have the following bound.

Lemma 14. *Suppose* S *is a split deletion set of size* k *of* G *and none of the Reduction Rules 3.1, 3.2 and 3.3 are applicable on the instance* (G,S)*. Then,* $|V(G)|$ *is* $\mathcal{O}(k + (2k+d)^d)$.

Proof. We will bound B, I_B and R separately in order to bound $V(G)$. We know that G has a connected vertex cover T of size at most $2k+d-1$. As B is the set of vertices of degree at least $2k+d$, $B \subseteq T$ and so $|B| \leq 2k+d-1$. Every vertex in R has degree at most $2k+d-1$. Therefore, as $T \cap R$ is a vertex cover of $G[R]$, $|E(G[R])|$ is $\mathcal{O}((2k+d-1)^2)$. Also, by the definition of I_B, every vertex in R has a neighbor in R and hence there are no isolated vertices in $G[R]$. Thus, $|R|$ is $\mathcal{O}((2k+d-1)^2)$. Finally, we bound the size of I_B.

As Reduction Rule 3.2 is not applicable, every vertex in $I_B \cap V(G-S)$ has degree at most $d-1$. For every set $B' \subseteq B$ of size at most $d-1$, there are at most $2k+d$ vertices in $I_B \cap V(G-S)$ which have B' as their neighborhood. Otherwise, Reduction Rule 3.3 would have been applied. Hence, there are at most $(2k+d) \cdot \binom{2k+(d-1)}{d-1}$ vertices in $I_B \cap V(G-S)$. Finally, as none of the reduction rules increases the size of the split deletion set S, we have $|I_B \cap S| \leq k$. Therefore, $|I_B|$ is $\mathcal{O}(k + (2k+d)^d)$. $\square$

It is clear from Lemma 9, 12 and 13 that all the above mentioned reduction rules can be implemented in polynomial time. This leads to a PSAKS for the problem as claimed.

Theorem 15. Connected Vertex Cover *parameterized by the size* k *of a split deletion set admits a time efficient PSAKS with* $\mathcal{O}(k + (2k + \lceil \frac{2\alpha-1}{\alpha-1} \rceil)^{\lceil \frac{2\alpha-1}{\alpha-1} \rceil})$ *vertices.*

Proof. Given $\alpha > 1$, we choose $d = \lceil \frac{2\alpha-1}{\alpha-1} \rceil$ and apply the reduction rules as long as they are applicable. Then, from Lemma 14, $|V(G)|$ is $\mathcal{O}(k + (2k + d)^d)$. □

4 Connected Vertex Cover parameterized by Clique Cover

In this section, we first show that some of the ideas from the previous section can be used to give a lossy kernel for Connected Vertex Cover when parameterized by the size of the clique cover. A clique cover of a graph is a partition of its vertex set such that each part induces a clique. We assume that we are given a partition of the vertex set of the input graph into cliques. For this parameterization, the corresponding parameterized minimization problem is defined in a similar way as defined for Connected Vertex Coverparameterized by split deletion set size.

Theorem 16. Connected Vertex Cover *parameterized by the size* k *of a clique cover admits a time efficient PSAKS with* $\mathcal{O}(k\lceil \frac{2\alpha-1}{\alpha-1} \rceil)$ *vertices.*

Proof. Without loss of generality, we assume that the graph has no isolated vertices as no minimum connected vertex cover contains them. Given $\alpha > 0$, let $d = \lceil \frac{2\alpha-1}{\alpha-1} \rceil$. Let $\mathcal{C}$ denote the set of k cliques in the clique cover of G. We bound the number of vertices in G by applying Reduction Rule 3.1 on each $C \in \mathcal{C}$. From Lemma 9, each application of this rule is α-safe. When this rule is no longer applicable, it follows that G has at most $k(d-1)$ vertices. This leads to the claimed result. □

Now, we modify a reduction known from [20] used in the hardness of Feedback Vertex Set with respect to the size of a clique cover to show that Connected Vertex Cover is $W[1]$-hard under the same parameterization. This makes this one of the few problems that are unlikely to be in FPT but have a lossy kernel.

Theorem 17. Connected Vertex Cover *parameterized by the size of a clique cover is* $W[1]$*-hard.*

Proof. We reduce the well known $W[1]$-hard problem, Independent Set parameterized by solution size, to our problem. A *non-separating independent set* of a graph G is an independent set I such that $V(G) \setminus I$ is a connected vertex cover of G. Let (G, k) be an instance of Independent Set. We construct a graph G' on the vertex set $\{(v, i) \mid v \in V(G), i \in [k]\} \cup \{x, y\}$. We add an edge between (v, i) and (u, j) if and only if $i = j$ or $u = v$ or $v \in N_G(u)$. Further, for every $w \in V(G') \setminus \{x\}$, we add the edge $\{w, x\}$ to G'. Note that G' has a clique cover of size $k+1$ as for all $j \in [k]$, $G'[\{(v, j) \mid v \in V(G)\}]$ is a complete graph and $\{x, y\}$ is a clique. It is easy to see that this construction can be done in polynomial time. We claim that G has an independent set of size k if and only if G' has a non-separating independent set of size $k+1$.

Suppose $S = \{v_1, \ldots, v_k\}$ is an independent set of size k in G. Define the set $S' \subseteq V(G')$ as $\{(v_1, 1), (v_2, 2), \ldots, (v_k, k), y\}$. Consider any two vertices (v_i, i) and (v_j, j) in S' such that $i \neq j$. As for each distinct $i, j \in [k]$, we have $\{v_i, v_j\} \notin E(G)$, it follows that there is no edge between (v_i, i) and (v_j, j). Also, by the construction of G', y is not adjacent to any vertex in G' other than x. Therefore, S' is an independent set of size $k+1$ in G'. As x is adjacent to every vertex in $G' - S'$, it follows that the deletion of S' does not disconnect G'. In other words, S' is a non-separating independent set of G'.

Conversely, suppose S' is a non-separating independent set of size $k+1$ in G'. Clearly, different vertices of S' have to come from different cliques of the clique cover of G'. We may assume that $y \in S'$ as if $x \in S'$, then $(S' \setminus \{x\}) \cup \{y\}$ is another non-separating independent set of size $k+1$. This is due to the facts that y is not adjacent to any vertex other than x and x is a universal vertex in G'. Define the set $S \subseteq V(G)$ as $\{v \mid (v,j) \in S' \setminus \{y\}\}$. Consider any two vertices (v,i) and (u,j) in S'. Then, $u \neq v$, $i \neq j$ and $v \notin N_G(u)$. Therefore, S is an independent set in G of size k. □

From Theorem 17, it follows that Connected Vertex Cover parameterized by $k+q$ is W[1]-hard where k is the size of a set S whose deletion results in a graph for which there exists a clique cover of size q. However, the problem is in XP as the algorithm in Theorem 4 can be easily generalized to solve Connected Vertex Cover in $O^*(2^k n^q)$ time where n is the number of vertices in the input graph. Further, by an easy adaptation of Theorem 16, the problem admits a PSAKS leading to the following result.

Theorem 18. *Given a graph G on n vertices and m edges, a set $S \subseteq V(G)$ such that $|S| \leq k$ and $G - S$ has a clique cover $\mathcal{Q}$ of size q and a positive integer ℓ, there is an algorithm that determines whether G has a connected vertex cover of size at most ℓ in $\mathcal{O}(2^k n^q(n+m))$ time. Further, the problem admits a time efficient PSAKS with $\mathcal{O}(k + q\lceil\frac{2\alpha-1}{\alpha-1}\rceil)$ vertices.*

Proof. Without loss of generality, we assume that the graph has no isolated vertices as no minimum connected vertex cover contains them. We also assume that the set S and the clique cover $\mathcal{Q}$ of $G - S$ are part of the input. First, let us describe the XP algorithm. Let X^* be a connected vertex cover of G of size at most ℓ. As at least $|Q| - 1$ vertices from each clique $Q \in \mathcal{Q}$ are contained in X^*, we guess $Y = X^* \cap (\bigcup_{Q \in \mathcal{Q}} Q)$. Then, we guess $Z = X^* \cap S$ such that $((\bigcup_{Q \in \mathcal{Q}} Q) \setminus Y) \cup (S \setminus Z)$ is an independent set. The number of choices for (Y, Z) is $\mathcal{O}(n^q 2^k)$. If $G[Y \cup Z]$ is not connected or has more than ℓ vertices, we skip to the next choice of (Y, Z). Otherwise, $Y \cup Z$ is the required solution. If none of the choices leads to a solution, we declare that G has no connected vertex cover of size at most ℓ. The overall running time of the algorithm is thus $\mathcal{O}(2^k n^q(n+m))$.

Now, we describe an α-approximate kernel for each $2 > \alpha > 1$. Given $\alpha > 0$, let $d = \lceil\frac{2\alpha-1}{\alpha-1}\rceil$. We bound the number of vertices in G by applying Reduction Rule 3.1 on each $Q \in \mathcal{Q}$. From Lemma 9, each application of this rule is α-safe and can be implemented in polynomial time. When this rule is no longer applicable, it follows that G has at most $k + q(d-1)$ vertices. This leads to the claimed result. □

5 Connected Vertex Cover parameterized by Cluster Deletion Set

In Section 3.1, we gave an $\mathcal{O}^*(2^k)$ time algorithm for Connected Vertex Cover parameterized by the size k of a clique deletion set. Here, we generalize this algorithm to solve Connected Vertex Cover in $\mathcal{O}^*(4^k)$ time where k is the size of a cluster deletion set. Observe that this parameter is smaller than the clique deletion set size. Further, we also describe a lossy kernel with respect to this parameterization. A classical polynomial kernel is unlikely as the size of a minimum cluster deletion set is at most the size of a minimum connected vertex cover. This is due to the fact that deleting a connected vertex cover from a graph results in a cluster graph in which every component is an isolated vertex.

5.1 An FPT Algorithm

Consider an instance (G, S, ℓ) of Connected Vertex Cover where S is a cluster deletion set. Let H denote the cluster graph $G - S$. Let X^* be a connected vertex cover of size at most ℓ (if one exists) that we are looking for. First, we guess the subset S' of S such that $S' = S \cap X^*$. If $S \setminus S'$ is not an independent set, we skip to the next choice of S'. Define the set F as $N(S \setminus S') \cap V(H)$. Initialize the set X to be S'. We also update ℓ to $\ell - |S'|$ and delete $S \setminus S'$ from G. In the latter steps of our algorithm, we will extend X to a connected vertex cover of G. We will now describe a sequence of reduction and branching rules to be applied. The rules are applied in the order stated and a rule is applied as long as it is applicable on the instance. That is, a rule is applied only when none of the preceding rules can be applied.

Let I denote the set of isolated vertices of H and $\mathcal{Q}$ denote the set of vertex sets of components of $H - I$. That is, I is an independent set and each element of $\mathcal{Q}$ is a clique in H. These sets are updated accordingly as and when the rules are applied. First, we apply the following preprocessing rule.

Preprocessing Rule 5.1. *If there is a clique $Q \in \mathcal{Q}$ with $N(Q) \cap X = \emptyset$ or a vertex $v \in F \cap I$ such that $N(v) \cap X = \emptyset$, then skip to the next choice of S'.*

The correctness of the first part of the rule follows from the fact that at least $|Q| - 1$ vertices from Q has to be in any vertex cover and any set of such vertices along with X cannot be extended to a connected subgraph by only adding vertices from $V(H)$. The correctness of the second part follows from the facts that F is forced into the solution and $G[X \cup \{v\}]$ cannot be extended to a connected subgraph by only adding vertices from $V(H)$. Computing I and $\mathcal{Q}$ can be done in $\mathcal{O}(|V(G - S)| + |E(G - S)|)$ time and identifying some vertex in I or some component from Q that satisfies the precondition as described can be done in $\mathcal{O}((|Q| + |I|) \cdot m)$ time. Thus, this preprocessing rule can be implemented in $\mathcal{O}(mn)$ time. Let F be partitioned into sets Y and Z defined as follows.

- $Y = \{v \in F \mid N(v) \cap X \neq \emptyset\}$, the set of vertices of F that have a neighbor in X.
- $Z = F \setminus Y$, the set of vertices of F that have no neighbor in X.

Observe that computing Y and Z takes $\mathcal{O}(|F| \cdot |S|)$ time. These sets are updated over the execution of the algorithm according to the current partial solution X. In particular, at any point of time, Z is the set of vertices in F that are not adjacent to any vertex in the current partial solution X. Our first reduction rule is as follows.

Reduction Rule 5.1. *Add $Y \cup Z$ to X, update ℓ to $\ell - |Z \cup Y|$ and delete Y from H.*

The correctness of this rule is justified by the fact that X^* must contain F. Further, this rule can be applied in $\mathcal{O}(|F||S| + |Y|)$ (which is $\mathcal{O}(nk)$ as $|F|, |Y| \leq n$ and $|S| \leq k$) time. Due to Preprocessing Rule 5.1, every vertex $v \in Z$ is in some clique Q in $\mathcal{Q}$. When Reduction Rule 5.1 is no longer applicable, we have $V(H) \cap F = Z$. For each $v \in Z$, let Q_v denote the clique in $\mathcal{Q}$ containing v. The next rule is the following.

Reduction Rule 5.2. *If there is a vertex $v \in Z$ with $|Q_v \cap Z| = |Q_v| - 1$, then add $Q_v \setminus Z$ to X, update ℓ to $\ell - 1$ and delete Q_v from H and Z.*

The correctness of this rule follows from the fact that there is exactly one vertex u in $Q_v \cap N(X \setminus Z)$. This vertex is forced into the solution since $Q_v \setminus \{u\} \subseteq X$ and $Q_v \setminus \{u\}$ has no neighbors in $X \setminus (Q_v \setminus \{u\})$. Determining the component of $G - S$ (i.e. Q_v) containing v can be done in $\mathcal{O}(n + m)$ time. Computing $Q_v \cap Z$ takes $\mathcal{O}(|Q_v| \log |Q_v| + |Z| \log |Z| + |Q_v| + |Z|)$

time. Finally, computing $Q_v \setminus Z$, adding it to X, updating the budget and deleting Q_v from H as well as Z takes $\mathcal{O}(n^2)$ time. Hence, this reduction rule can be applied in $\mathcal{O}(n^2)$ time. Now, for any clique $Q \in \mathcal{Q}$, there are at least two vertices that are not in Z. For a vertex $v \in V(H)$, let $\mathcal{C}(v) = \{A \in G[X] \mid A \text{ is a component of } G[X] \text{ and } N_G(v) \cap V(A) \neq \emptyset\}$. The next rule is the following.

Reduction Rule 5.3. *If there is a clique* Q *in* $\mathcal{Q}$ *that has two vertices* $u, v \notin Z$ *with* $\mathcal{C}(u) \subseteq \mathcal{C}(v)$*, then update* X *to* $X \cup (Q \setminus \{u\})$ *and reduce* ℓ *by the number of vertices added to* X*. Delete* Q *from* H *and* Z*.*

If there exists an optimal connected vertex cover X that does not contain v, then $u \in X$ and $X' = (X \setminus \{u\}) \cup \{v\}$ is also a connected vertex cover of G. This justifies the correctness of the rule. Identifying a clique $Q \in \mathcal{Q}$ containing two vertices u, v satisfying the required property, deleting Q from H and Z and updating ℓ can be done in $\mathcal{O}(n^3)$ time. The next rule is a branching rule that is applied when H has a triangle or a larger clique Q with vertices not in Z. Further, for each $v \in Q \setminus Z$, $|\mathcal{C}(v)| \geq 1$ and for every $u, v \in Q \setminus Z$, $\mathcal{C}(u) \not\subseteq \mathcal{C}(v)$.

Branching Rule 5.1. *If there is a triangle* $\{u, v, w\}$ *in* H *with* $u, v, w \notin Z$*, then branch into adding* u, v *or* u, w *or* w, v *into* X*. In each of the branches update* ℓ *to* $\ell - 2$*.*

To apply this rule, we need to determine if there is a clique Q with at least three vertices such that for each pair of vertices $u, v \in Q \setminus Z$, we have $\mathcal{C}(u) \not\subseteq \sqsubseteq$. This can be achieved in $\mathcal{O}(n^4)$ time. The branches in this rule are clearly exhaustive as any vertex cover contains at least two vertices from a triangle. The reason for handling such special triangles will be apparent during the application of subsequent rules. When none of the rules described so far is applicable, every clique $Q \in \mathcal{Q}$ has exactly 2 vertices not in Z. In particular, if $Q \cap Z = \emptyset$, then $|Q| = 2$.

Reduction Rule 5.4. *If there is an edge* $\{u, v\}$ *with* $u, v \notin Z$ *in* H *and* $|\mathcal{C}(u)| = 1$ *and* $|\mathcal{C}(v)| \geq 1$*, then update* X *to* $X \cup (Q \setminus \{u\})$ *and reduce* ℓ *by the number of vertices added to* X *where* Q *is the clique in* $\mathcal{Q}$ *that contains* u *and* v*. Delete* $Q \setminus \{u\}$ *from* H *and* Z*. Add* u *to* I*.*

If there exists an optimal connected vertex cover X that does not contain v, then $u \in X$ and $X' = (X \setminus \{u\}) \cup \{v\}$ is also a vertex cover of G. Further, $G[X']$ is connected as the only vertex in $Q \cap X$ that is adjacent to a vertex in $X \setminus \{u\}$ is u and $|\mathcal{C}(u)| = 1$. Note that we crucially use the property that Q has exactly two vertices adjacent to a vertex in $X \setminus \{u, v\}$ which is achieved by the application of Branching Rule 5.1. This justifies the correctness of the rule. Implementing this reduction rule requires us to check whether there are two vertices outside $u, v \notin Z$ that are adjacent in H satisfying $|\mathbb{C}(u)| = 1$ and $|\mathcal{C}(v)| \geq 1$. This can be done in $\mathcal{O}(n^3)$ time. The next rule is a branching rule that is applied until X is a vertex cover (not necessarily connected) of G.

Branching Rule 5.2. *If there is an edge* $\{u, v\}$ *in* H*, then in one branch we add* u *into* X *and in other branch we add* v *into* X*. In both branches, update* ℓ *to* $\ell - 1$ *and delete the vertex added to* X *from* H*.*

When this rule is applied on an edge $\{u, v\}$, we have $|\mathcal{C}(x)|, |\mathcal{C}(y)| \geq 2$ and $\{x, y\} \cap Z = \emptyset$. The branches are exhaustive as any vertex cover contains x or y. At this point, when none of the described rules (reduction and branching) is applicable, we have a vertex cover X of G. That is, $V(H)$ is an independent set. However, X may not necessarily induce a connected subgraph. Let G' be the graph obtained from $G[V(H) \cup X]$ by contracting each component of $G[X]$ into a single vertex. G' is bipartite and has a bipartition $(V(H), \widehat{X})$ where $\widehat{X}$ is the set of vertices of G' corresponding to the set of components of $G[X]$. The problem is now to find a Steiner tree of G' that contains $\widehat{X}$ which can be solved in $\mathcal{O}(2^{|\widehat{X}|} \cdot \ell(m + n))$ time using Lemma 1. This completes the description of the algorithm leading to the following result.

Theorem 19. *Given a graph* G*, a cluster deletion set* S *and a positive integer* ℓ*, there is an algorithm that determines whether* G *has a connected vertex cover of size at most* ℓ *in* $\mathcal{O}(4^{|S|} \cdot \ell(m+n) \cdot n^4)$ *time.*

Proof. Given an instance (G, S, ℓ), we first guess a subset S' of S that is contained in the solution. Then, we apply the reduction rules and branching rules in the order stated. We reiterate that each rule is applied as long as it is applicable. Also, a rule is applied only when none of the earlier rules is applicable. Let X be the partial solution that we are trying to extend to a connected vertex cover. Initially X is S'. The measure that we use to bound the running time is the number #comp($G[X]$) of components of $G[X]$ which is at most $|S'|$.

On applying either of the branching rules, it is clear that #comp($G[X]$) drops in all the branches by at least one. We will now show that the reductions rules do not increase the measure. When Reduction Rule 5.2 is applied, Q_v has a neighbor in X due to Preprocessing Rule 5.1. Similarly, when Reduction Rule 5.3 or 5.4 is applied, $Q \setminus \{u\}$ has a neighbor in X. Hence, these three rules do not increase #comp($G[X]$). Now consider Reduction Rule 5.1. Adding Y to X do not increase #comp($G[X]$) while adding Z to X will definitely increase it. However, if Z is non-empty, then in subsequent rules, some neighbor of Z which is also adjacent to some vertex in (current) X is added to the solution. Consider a vertex $z \in Z$ on which Reduction Rule 5.2 is applicable. Then, we add a neighbor z' of z that is adjacent to X. Therefore, the measure does not increase. In fact, the measure first increases by 1 (due to addition of z to X) and then decreases by at least 1 (due to addition of z' to X). Suppose Reduction Rule 5.2 is not applicable on a vertex $z \in Z$. Then, Q_z has at least 2 vertices not in Z. If Reduction Rule 5.3 is applicable on Q_z, then adding z into X does not increase the measure. Let us consider the case when Reduction Rule 5.3 is not applicable on Q_z. Then, either Branching Rule 5.1 or Reduction Rule 5.4 or Branching Rule 5.2 is applicable to vertices of Q_z. In any case, as we add a neighbor of z that is adjacent to X, the measure does not increase.

When none of the reduction and branching rules is applicable, we have a set X that is a vertex cover of G. That is, at each leaf of the recursion tree, we have a vertex cover X of G that needs to be connected by adding a minimum number of vertices from $V(H)$. We construct the bipartite graph G' with bipartition $(V(H), \widehat{X})$ as described earlier. Here, $\widehat{X}$ is the set of vertices of G' corresponding to the set of components of $G[X]$. Observe that #comp($G[X]$) is at most $|S'|$. Then, the time taken at each leaf of the recursion tree is upper bounded by the time taken to find a Steiner tree of G' that contains $\widehat{X}$ which is $\mathcal{O}(2^{|\widehat{X}|} \cdot \ell(m+n))$ from Lemma 1. Let $|S'| = i$. If a leaf is at depth j where $j \leq i$, it follows that the #comp($G[X]$) is at most $i - j$. Then, the Steiner tree algorithm at this leaf runs in $\mathcal{O}^*(2^{i-j})$ time.

Let $T(j)$ denote the number of leaves of the search tree at depth j. It is easy to verify that the search tree is a ternary tree as there are at most three branches in any rule. Thus, the number of leaves at depth j is at most 3^j. Therefore, the running time (ignoring polynomial factors) of the algorithm is upper bounded by the following value. Let k denote $|S|$.

$$\sum_{i=0}^{k} \binom{k}{i} \sum_{j=0}^{i} 3^j 2^{i-j} \leq \sum_{i=0}^{k} i \binom{k}{i} 3^i = 3 \sum_{i=1}^{k} k \binom{k-1}{i-1} 3^{i-1} = 3k \cdot 4^{k-1}$$

Thus, the running time of the algorithm is $\mathcal{O}(4^k \cdot \ell(m+n) \cdot n^4)$ as each of the rules can be implemented in $\mathcal{O}(n^4)$ time. □

A degree-i modulator is a set of vertices whose deletion results in a graph with maximum degree at most i. As a degree-1 modulator is also a cluster deletion set, it follows that

Connected Vertex Cover can be solved in $O^*(4^{|S|})$ time when given a degree-1 modulator S as part of input. However, observe that this algorithm runs in $\mathcal{O}^*(3^k)$ time as Branching Rule 5.1 is never applicable (and so the search tree is a binary tree). Thus, we have the following result.

Theorem 20. *Given a graph G, a degree-1 modulator S and a positive integer ℓ, there is an algorithm that determines whether G has a connected vertex cover of size at most ℓ in $\mathcal{O}(3^{|S|} \cdot \ell(m+n) \cdot n^4)$ time.*

It is well known that the treewidth (tw) of a graph is not larger than the size of its minimum vertex cover, and hence of its minimum connected vertex cover. Therefore, a naive dynamic programming routine over a tree decomposition of width tw solves Connected Vertex Cover in $\mathcal{O}^*(2^{tw \log tw})$ time [23]. The running time bound has been improved to $\mathcal{O}^*(2^{\mathcal{O}(tw)})$ using algebraic techniques [2]. If the maximum degree of a graph G is upper bounded by 2, then every component of G is an induced path or an induced cycle. Hence, the treewidth of G is at most 2. Therefore, if G has a degree-2 modulator of size at most k, then the treewidth of G is at most $k+2$. This shows that Connected Vertex Cover is FPT when parameterized by the size of a degree-2 modulator. As Connected Vertex Cover is NP-hard on graphs with maximum degree at most 4, it is clear that Connected Vertex Cover when parameterized by the size of a degree-4 modulator is para-NP-hard (NP-hard for fixed values of parameter). This observation doesn't immediately carry over when parameterized by the size of a degree-3 modulator as Connected Vertex Cover is polynomial time solvable in sub-cubic graphs [24]. Nevertheless, Connected Vertex Cover is para-NP-hard even when parameterized by the size of a degree-3 modulator. This is due to the fact that Vertex Cover in sub-cubic graphs is NP-complete [1, 15] and Vertex Cover reduces to Connected Vertex Cover by just adding a universal vertex (which is a degree-3 modulator) to the graph.

Theorem 21. Connected Vertex Cover *parameterized by the size of a degree-i modulator is* FPT *when $i \leq 2$ and para-*NP*-hard for $i \geq 3$.*

It is intriguing that though Connected Vertex Cover is polynomial-time solvable on sub-cubic graphs, it is NP-hard on graphs that are just one vertex away from being a sub-cubic graph. Note that the related Feedback Vertex Set problem is also solvable in polynomial-time on sub-cubic graphs, but it is a major open problem whether it is polynomial-time solvable on graphs that is just one vertex away from a sub-cubic graph.

5.2 A Lossy Kernel

In this section, we give a PSAKS for Connected Vertex Cover parameterized by the size of a cluster deletion set. We formally define the parameterized minimization problem as follows.

$$\mathsf{CVC}((G,S),k,T) = \begin{cases} -\infty & \text{if } |S| > k \text{ or some component of } G-S \text{ is not a clique} \\ \infty & \text{if } T \text{ is not a connected vertex cover} \\ |T| & \text{otherwise} \end{cases}$$

We now prove the following main result of this section.

Theorem 22. Connected Vertex Cover *parameterized by the size k of a cluster deletion set admits a time efficient PSAKS with $\mathcal{O}(k^2 + \lceil \frac{2\alpha-1}{\alpha-1} \rceil \cdot \frac{k}{\alpha-1} + \lceil \frac{\alpha}{\alpha-1} \rceil \cdot k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$ vertices.*

Proof. Let G be a connected graph and S denote its cluster deletion set of size at most k. Given $\alpha > 1$, define $\epsilon = \alpha - 1$. Let H denote the cluster graph $G-S$ and F_0 be the set of isolated vertices in H. Let $F_1 = V(H) \setminus F_0$ and t denote the number of components in $G[F_1]$. Let $C_1, \cdots, C_t$ denote the components of $G[F_1]$. Let $d_1 = \lceil \frac{2\alpha-1}{\alpha-1} \rceil$ and $d_2 = \lceil \frac{\alpha}{\alpha-1} \rceil$. Define the set T as follows. We first initialize T to S. Then, for each $i \in [t]$, we add a set X_i of $|V(C_i)|-1$ vertices of C_i such that $N_G(X_i) \cap S \neq \emptyset$ to T. Such a set always exists as G is connected. Now, T is a vertex cover of G and $|T| = |S| + \sum\limits_{i=1}^{t}(|V(C_i)|-1)$. Further, $\#\text{comp}(G[T]) = \#\text{comp}(G[S])$. If $G[T]$ is not connected, then we add at most $|S|-1$ vertices from $V(G) \setminus T$ to T so that $G[T]$ becomes connected. Once again, such a set exists as G is connected. Thus, we have $|T| < 2k + \sum\limits_{i=1}^{t}(|V(C_i)|-1)$. We know that $\mathsf{OPT}((G,S),k) \geq \sum\limits_{i=1}^{t}(|V(C_i)|-1)$ as each $V(C_i)$ is a clique and any vertex cover excludes at most one vertex from any clique. Therefore, we have that $|T| < 2k + \mathsf{OPT}((G,S),k)$. If $t \geq \frac{2k}{\epsilon}$, then the reduction algorithm outputs a constant size instance (G', S'). Since $\mathsf{OPT}((G,S),k) \geq t$, we have $k \leq \frac{\epsilon}{2}\mathsf{OPT}((G,S),k)$ and it follows that $|T|$ is at most $(1+\epsilon)\mathsf{OPT}((G,S),k)$. Equivalently, $\mathsf{CVC}((G,S),k,T) \leq (1+\epsilon)\mathsf{OPT}((G,S),k)$. The solution lifting algorithm simply returns T which is obtained in polynomial time.

On the other hand, suppose $t < \frac{2k}{\epsilon}$. For each $i \in [t]$, we apply Reduction Rule 3.1 with $d = d_1$ to bound the number of vertices in C_i by d_1. From Lemma 9, we know that Reduction Rule 3.1 is α-safe. When this rule is no longer applicable, we have at most $d_1 \frac{2k}{\epsilon}$ vertices in F_1. Now, it remains to bound the number of vertices in F_0. Observe that any optimal connected vertex cover must contain any vertex $x \in S$ with $|N_G(x) \cap F_0| \geq 2k$. This is because, if some optimal connected vertex cover T^* excludes x, then $|T^*| \geq 2k + \sum\limits_{i=1}^{t}(|V(C_i)|-1)$. However, T is a connected vertex cover with $|T| < 2k + \sum\limits_{i=1}^{t}(|V(C_i)|-1)$ leading to a contradiction. Let $S_1 = \{x \in S \mid |N_G(x) \cap F_0| \geq 2k\}$. We first partition F_0 into $A_0 = \{v \in F_0 \mid N_G(v) \subseteq S_1\}$ and $B_0 = F_0 \setminus A_0$. Then, we apply Reduction Rule 3.2 to vertices in A_0 with $d = d_2$ and Reduction Rule 3.3 to vertices in A_0 with $d = 0$. Reduction Rule 3.2 is α-safe and Reduction Rule 3.3 is 1-safe from Lemmas 12 and 13.

Suppose none of the described rules is applicable on the instance (G,S). We will show that $|V(G)| = \mathcal{O}(k^2 + d_2 \cdot k^{d_2} + d_1 \cdot k)$. The set $V(G)$ is partitioned into sets S, F_1, A_0 and B_0. As G is connected, G has no isolated vertex. As Reduction Rule 3.1 is not applicable, each component in $G[F_1]$ has at most d_1 vertices. Thus, $|V(F_1)| \leq \frac{2k}{\epsilon} \cdot d_1$. For any vertex $v \in F_0$, there is a vertex $x \in S$ such that $\{x, v\} \in E(G)$. Since the reduction rules described are not applicable, every vertex $x \in A_0$ has at most $d_2 - 1$ neighbors in S_1. Further, for every $X \in \binom{S_1}{\leq d_2-1}$, there are at most $2k$ vertices from A_0 such that each of those $2k$ vertices has X as its neighborhood in G. Therefore, $|A_0| \leq 2k \cdot \sum\limits_{i=1}^{d_2-1} \binom{k}{i} \leq 2(d_2-1)k^{d_2}$ as $|S_1| \leq k$. As G is connected, each vertex in B_0 has a neighbor in $S \setminus S_1$. Since, the degree of any vertex in $S \setminus S_1$ is at most $2k$, it follows that $|B_0| \leq 2k^2$. Thus, $|V(G)| = |S| + |F_1| + |A_0| + |B_0|$ is at most $k + \frac{2k}{\epsilon} \cdot d_1 + 2(d_2-1)k^{d_2} + 2k^2$ which is $\mathcal{O}(k^2 + \lceil \frac{2\alpha-1}{\alpha-1} \rceil \cdot \frac{k}{\alpha-1} + \lceil \frac{\alpha}{\alpha-1} \rceil \cdot k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$. $\square$

As the deletion of a degree-1 modulator results in a graph in which every component has at most 2 vertices, we have the following result as a consequence of Theorem 22.

Corollary 23. Connected Vertex Cover *parameterized by the size* k *of a degree-*1 *modulator admits a time efficient PSAKS with* $\mathcal{O}(k^2 + \frac{k}{\alpha-1} + \lceil \frac{\alpha}{\alpha-1} \rceil \cdot k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$ *vertices.*

6 Concluding Remarks

We have given lossy kernels and studied the parameterized complexity statuses of Connected Vertex Cover parameterized by split deletion set size, clique cover number and cluster deletion set size. Our FPT running times (for Connected Vertex Cover parameterized by split deletion set and cluster deletion set) have slight gaps between upper bounds and lower bounds (based on well-known conjectures), and tightening the bounds is an interesting open problem. A more general direction is to explore the parameterized and (lossy) kernelization complexity of Connected Vertex Cover in the parameter ecology program.

It is well known that the treewidth (tw) of a graph is at most one more than the size of a minimum feedback vertex set (a set of vertices whose removal results in a forest). This observation along with the FPT algorithm for Connected Vertex Cover parameterized by treewidth immediately implies that Connected Vertex Cover is FPT when parameterized by the size of a feedback vertex set.

Once again by using the algorithm for bounded treewidth graphs, we can show that Connected Vertex Cover is FPT when parameterized by chordal deletion set size [1]. As forests, split graphs and cluster graphs are chordal, the size of a minimum chordal deletion set is at most the size of a minimum feedback vertex set or split deletion set or cluster deletion set. However, the lossy kernelization status of this problem remains open even when the chordal deletion set is a feedback vertex set.

Figure 2 gives a partial landscape of the (parameterized) complexity of Connected Vertex Cover under various structural parameters. Designing (lossy) kernels (or proving impossibility results) for those whose status is unknown is an interesting future direction.

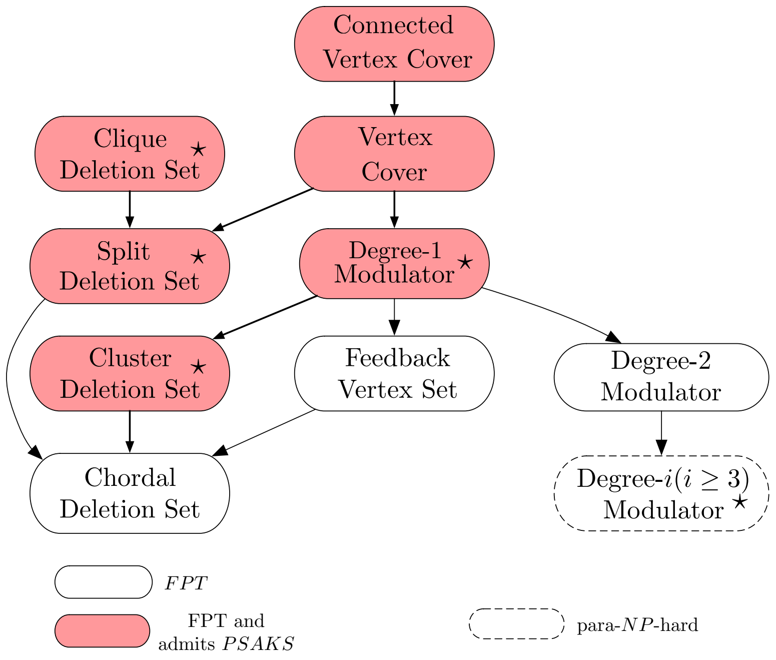

Figure 2: Ecology of Parameters for Connected Vertex Cover. The parameter values are the minimum possible for a given graph. An arrow from parameter $\mathtt{x}$ to parameter $\mathtt{y}$ means $\mathtt{y} \leq \mathtt{x}$. Parameters marked $\star$ are the ones considered in this paper.

[1] A chordal graph is a graph in which every induced cycle is a triangle and a chordal deletion set is a subset of vertices whose deletion results in a chordal graph.

References

[1] P. Alimonti and V. Kann. Some APX-Completeness Results for Cubic Graphs. *Theoretical Computer Science*, 237(1-2):123–134, 2000.

[2] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic Single Exponential Time Algorithms for Connectivity Problems parameterized by Treewidth. *Information and Computation*, 243:86–111, 2015.

[3] H. L. Bodlaender and B. M. P. Jansen. Vertex Cover Kernelization Revisited: Upper and Lower Bounds for a Refined Parameter. *Theory of Computing Systems*, 63(2):263–299, 2013.

[4] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM Journal of Discrete Mathematics*, 28(1):277–305, 2014.

[5] A. Boral, M. Cygan, T. Kociumaka, and M. Pilipczuk. A Fast Branching Algorithm for Cluster Vertex Deletion. *Theory of Computing Systems*, 58(2):357–376, 2016.

[6] J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further Observations and Further Improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[7] M. Cygan. Deterministic Parameterized Connected Vertex Cover. In *Proceedings of the 13th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 95–106, 2012.

[8] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlstrm. On Problems as Hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41:1–41:24, 2016.

[9] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms.* Springer, 2015.

[10] M. Cygan and M. Pilipczuk. Split Vertex Deletion meets Vertex Cover: New Fixed-Parameter and Exact Exponential-Time Algorithms. *Information Processing Letters*, 113(5-6):179–182, 2013.

[11] R. Diestel. *Graph Theory.* Springer, Graduate Text in Mathematics, 2012.

[12] M. Dom, D. Lokshatanov, and S. Saurabh. Kernelization Lower Bounds through Colors and IDs. *ACM Transaction on Algorithms*, 11(2):13:1–13:20, 2014.

[13] B. Escoffier, L. Gourvs, and J. Monno. Complexity and Approximation Results for the Connected Vertex Cover problem in Graphs and Hypergraphs. *Journal of Discrete Algorithms*, 8(1):36–49, 2010.

[14] F. Fomin and D. Kratsch. *Exact Exponential Algorithms.* Springer-Verlag, 2010.

[15] M. R. Garey and D. S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[16] M. C. Golumbic. *Algorithmic Graph Theory for Perfect Graphs.* Springer, 2004.

[17] B. M. P. Jansen. *The Power of Data Reduction: Kernels for Fundamental Graph Problems.* PhD thesis, Utrecht University, The Netherlands, 2013.

[18] B. M. P. Jansen, M. R. Fellows, and F. A. Rosamond. Towards fully Multivariate Algorithmics: Parameter Ecology and the Deconstruction of Computational Complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.

[19] B. M. P. Jansen and S. Kratsch. Data Reduction for Coloring Problems. *Information and Computation*, 231:70–88, 2013.

[20] B. M. P. Jansen, V. Raman, and M. Vatshelle. Parameter Ecology for Feedback Vertex Set. *Tsinghua Science and Technology*, 19(4):387–409, 2014.

[21] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster Parameterized Algorithms Using Linear Programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014.

[22] D. Lokshtanov, F. Panolan, M. S. Ramanujan, and S. Saurabh. Lossy Kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 224–237, 2017.

[23] H. Moser. Exact Algorithms for Generalizations of Vertex Cover. Master's thesis, Institut für Informatik, Friedrich-Schiller-Universität, 2005.

[24] S. Ueno, Y. Kajitani, and S. Gotoh. On the Nonseparating Independent Set problem and Feedback Set problem for Graphs with no Vertex Degree exceeding Three. *Discrete Mathematics*, 72:355–360, 1988.

[25] B. Y. Wu. A Measure and Conquer based Approach for the Parameterized Bounded Degree-one Vertex Deletion. In *Proceedings of the 21st International Computing and Combinatorics Conference (COCOON)*, pages 469–480, 2015.