A Faster FPT Algorithm and a Smaller Kernel for Block Graph Vertex Deletion

Akanksha Agrawal[1(✉)], Sudeshna Kolay[2], Daniel Lokshtanov[1], and Saket Saurabh[1,2]

[1] University of Bergen, Bergen, Norway
{akanksha.agrawal,daniello}@uib.no
[2] Institute of Mathematical Sciences, Chennai, India
{skolay,saket}@imsc.res.in

Abstract. A graph G is called a *block graph* if every maximal 2-connected component of G is a clique. In this paper we study the Block Graph Vertex Deletion from the perspective of fixed parameter tractable (FPT) and kernelization algorithms. In particular, an input to Block Graph Vertex Deletion consists of a graph G and a positive integer k, and the objective to check whether there exists a subset $S \subseteq V(G)$ of size at most k such that the graph induced on $V(G) \setminus S$ is a block graph. In this paper we give an FPT algorithm with running time $4^k n^{\mathcal{O}(1)}$ and a polynomial kernel of size $\mathcal{O}(k^4)$ for Block Graph Vertex Deletion. The running time of our FPT algorithm improves over the previous best algorithm for the problem that runs in time $10^k n^{\mathcal{O}(1)}$, and the size of our kernel reduces over the previously known kernel of size $\mathcal{O}(k^6)$. Our results are based on a novel connection between Block Graph Vertex Deletion and the classical Feedback Vertex Set problem in graphs without induced C_4 and $K_4 - e$. To achieve our results we also obtain an algorithm for Weighted Feedback Vertex Set running in time $3.618^k n^{\mathcal{O}(1)}$ and improving over the running time of previously known algorithm with running time $5^k n^{\mathcal{O}(1)}$.

1 Introduction

Deleting the minimum number of vertices from a graph such that the resulting graph belongs to a family $\mathcal{F}$ of graphs, is a measure on how close the graph is to the graphs in the family $\mathcal{F}$. In the problem of vertex deletion, we ask whether we can delete at most k vertices from the input graph G such that the resulting graph belongs to the family $\mathcal{F}$. Lewis and Yannakakis [12] showed that for any non-trivial and hereditary graph property Π on induced subgraphs, the vertex deletion problem is NP-complete. Thus these problems have been subjected to intensive study in algorithmic paradigms that are meant for coping with NP-completeness [7,8,13,14]. These paradigms among others include applying

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 306992.

E. Kranakis et al. (Eds.): LATIN 2016, LNCS 9644, pp. 1–13, 2016.
DOI: 10.1007/978-3-662-49529-2_1

restrictions on inputs, approximation algorithms and parameterized complexity. The focus of this paper is to study one such problem from the viewpoint of parameterized algorithms.

Given a family $\mathcal{F}$, a typical parameterized vertex deletion problem gets as an input an undirected graph G and a positive integer k and the goal is to test whether there exists a vertex subset $S \subseteq V(G)$ of size at most k such that $G \setminus S \in \mathcal{F}$. In the parameterized complexity paradigm the main objective is to design an algorithm for the vertex deletion problem that runs in time $f(k) \cdot n^{\mathcal{O}(1)}$, where $n = |V(G)|$ and f is an arbitrary computable function depending only on the parameter k. Such an algorithm is called an FPT algorithm and such a running time is called FPT running time. We also design a polynomial time preprocessing algorithm that reduces the given instance to an equivalent one with size as small as possible. This is mathematically modelled by the notion of *kernelization*. A parameterized problem is said to admit a $h(k)$-*kernel* if there is a polynomial time algorithm (the degree of the polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance to an equivalent instance with size upper bounded by $h(k)$. In other words, let (x, k) be the input instance and (x', k') be the reduced instance. Then, $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$. Also, $|x'|, k' \leq h(k)$. If the function $h(k)$ is polynomial in k, then we say that the problem admits a polynomial kernel. For more background, the reader may refer to the following monograph [6].

A graph G is known as a *block graph* if every maximal 2-connected component in G is a clique. Equivalently, we can see a block graph as a graph obtained by replacing each edge in a forest by a clique. A *chordal graph* is a graph which has no induced cycles of length at least four. An equivalent characterisation of a block graph is a chordal graph with no induced $K_4 - e$ [2,9]. The class of block graphs is the intersection of the chordal and distance-hereditary graphs [9].

In this paper, we consider the problem which we call Block Graph Vertex Deletion (BGVD). Here, as an input we are given a graph G and an integer k, and the question is whether we can find a subset $S \subseteq V(G)$ of size at most k such that $G \setminus S$ is a block graph. The NP-hardness of the BGVD problem follows from [12].

Block Graph Vertex Deletion (BGVD) **Parameter:** k
Input: An undirected graph $G = (V, E)$, and a positive integer k
Question: Is there a set $S \subseteq V$, of size at most k, such that $G \setminus S$ is a block graph?

Kim and Kwon [10] gave an FPT algorithm with running time $\mathcal{O}^{\star}(10^k)$ and a kernel of size $\mathcal{O}(k^6)$ for the BGVD problem. In this paper we improve both these results via a novel connection to Feedback Vertex Set problem.

Our Results and Methods. We start by giving the results we obtain in this article and then we explain how we obtain these results. Our three main results are:

Theorem 1. BGVD *has an* FPT *algorithm running in time* $\mathcal{O}^\star(4^k)$.

Theorem 2. BGVD *admits a factor four approximation algorithm.*

Theorem 3. BGVD *has a kernel of size* $\mathcal{O}(k^4)$.

Our two of the theorems improve both the results in [10]. That is, the running time of our FPT algorithm improves over the previous best algorithm for the problem that runs in time $10^k n^{\mathcal{O}(1)}$, and the size of our kernel reduces over the previously known kernel of size $\mathcal{O}(k^6)$.

Our results are based on a connection between the WEIGHTED-FVS and BGVD problems. In particular we show that if the given input graph does not have induced four cycles or diamonds ($K_4 - e$) then we can construct an auxiliary bipartite graph and solve WEIGHTED-FVS on it. This results in a faster FPT algorithm for BGVD. In the algorithm that we give for the BGVD problem, as a sub-routine we use the algorithm for the WEIGHTED-FVS problem. For obtaining a better polynomial kernel for BGVD, most of our Reduction Rules are same as those used in [10]. On the way to our result we also design a factor four approximation algorithm for BGVD.

Finally, we talk about WEIGHTED-FVS. For which, we also design a faster algorithm than known in the literature. The FEEDBACK VERTEX SET problem is one of the most well studied problems. Given an undirected graph $G = (V, E)$ and a positive integer k, the problem is to decide whether there is a set $S \subseteq V$ such that $G \setminus S$ is a forest. Thus, S is a vertex subset that intersects with every cycle of G. In the parameterized complexity setting, FEEDBACK VERTEX SET parameterized by k, has an FPT algorithm. The best known FPT algorithm runs in time $\mathcal{O}^\star(3.618^k)$ [4,11]. The problem also admits a kernel on $\mathcal{O}(k^2)$ vertices [15]. Another variant of FEEDBACK VERTEX SET that has been studied in parameterized complexity is WEIGHTED FEEDBACK VERTEX SET, where each vertex in the graph has some rational number as its weight.

WEIGHTED-FVS **Parameter:** k
Input: An undirected graph $G = (V, E)$, a weight function $w : V \to \mathbb{Q}$, and a positive integer k
Output: The minimum weighted set $S \subseteq V$ of size at most k, such that $G \setminus S$ is a forest.

WEIGHTED-FVS is known to be in FPT with an algorithm of running time $5^k n^{\mathcal{O}(1)}$ [3]. We obtain a faster FPT algorithm for WEIGHTED-FVS. This algorithm uses, as a subroutine, the algorithm for solving WEIGHTED-MATROID PARITY [16]. In fact, this algorithm is very similar to the algorithm for FEEDBACK VERTEX SET given in [4,11]. Thus, our final new result is the following theorem.

Theorem 4 [$\star$]**.** WEIGHTED-FVS *has an* FPT *algorithm running in time* $\mathcal{O}^\star(3.618^k)$.

Due to paucity of space, results stated without proof in the short version are marked with [$\star$]. These proofs can be found in the full version of the paper.

2 Preliminaries

We start with some basic definitions and terminology from graph theory and algorithms. We also establish some of the notation that will be used in this paper.

We will use the $\mathcal{O}^\star$ notation to describe the running time of our algorithms. Given $f : \mathbb{N} \to \mathbb{N}$, we define $\mathcal{O}^\star(f(n))$ to be $\mathcal{O}(f(n) \cdot p(n))$, where $p(\cdot)$ is some polynomial function. That is, the $\mathcal{O}^\star$ notation suppresses polynomial factors in the running-time expression. We denote the set of rational numbers by $\mathbb{Q}$.

Graphs. A graph is denoted by $G = (V, E)$, where V and E are the vertex and edge sets, respectively. We also denote the vertex set and edge set of G by $V(G)$ and $E(G)$, respectively. All the graphs that we consider are finite graphs, possibly having loops and multi-edges. For any non-empty subset $W \subseteq V(G)$, the subgraph of G induced by W is denoted by $G[W]$; its vertex set is W and its edge set consists of all those edges of $E(G)$ with both endpoints in W. For $W \subseteq V(G)$, by $G \setminus W$ we denote the graph obtained by deleting the vertices in W and all edges which are incident to at least one vertex in W.

For a graph G, we denote the degree of vertex v in G by $d_G(v)$. A vertex $v \in V(G)$ is called as a *cut vertex* if the number of connected components in $G \setminus \{v\}$ is more than the number of connected components in G. For a vertex $v \in V(G)$, the neighborhood of v in G is the set $N_G(v) = \{u | (v, u) \in E(G)\}$. We drop the subscript G from $N_G(v)$, whenever the context is clear. Two vertices $u, v \in V(G)$ are called *true-twins* in G if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. For $A \subset V(G)$, an A-path in G is a path with at least one edge, whose end vertices are in A and all the internal vertices are from $V(G) \setminus A$.

A weighted undirected graph is a graph $G = (V, E)$, with a weight function $w : V(G) \to \mathbb{Q}$. For a subset $X \subseteq V(G)$, $w(X) = \sum_{v \in X} w(v)$.

A *feedback vertex set* is a subset $S \subseteq V(G)$ such that $G \setminus S$ is a forest. A minimum weight feedback vertex set of a weighted graph G is a subset $X \subseteq V(G)$, such that $G \setminus X$ is a forest and $w(X)$ is minimum among all possible weighted-fvs in G. In a graph with vertex weights, an FVS is called a weighted feedback vertex set (weighted-fvs). Similarly, for a given positive integer k, a minimum weighted-fvs of size k is a subset $X \subseteq V(G)$ such that $|X| \leq k$, $G \setminus X$ is a forest and $w(X)$ is minimum among all possible weighted-fvs in G that are of size at most k. Given a graph G and a vertex subset $S \subseteq V(G)$, we say that S is a *block vertex deletion set* if $G \setminus S$ is a block graph.

A maximal 2-connected subgraph of a graph G is called a *block*. By $K_4 - e$ we denote the graph obtained by removing an edge e from a complete graph on 4 vertices. For a graph G, let V_c denote the set of cut vertices of G, and $\mathcal{B}$ the set of its blocks. We then have a natural bipartite graph F on $V_c \cup \mathcal{B}$ formed by the edges (v, B) if and only if $c \in V(B)$. Note that for a block graph G, F is a forest [5]. The bipartite graph F is called as the block forest of G. We will arbitrarily root F at some vertex $B \in V(F)$.

A leaf block of a block graph G is a maximal 2-connected component with at most one cut vertex. For a maximal 2-connected component C in G a vertex $v \in V(C)$ is called as an *internal vertex* if v is not a cut vertex in G.

We refer the reader to [5] for details on standard graph theoretic notation and terminology we use in the paper.

3 FPT Algorithm for Block Graph Vertex Deletion

In this section, we present an FPT algorithm for the BGVD problem. First, we look at the special case, when the input graph does not have any small obstructions in the form of D_4's and C_4's. Here, $D_4 = K_4 - e$. We show that, in this case, BGVD reduces to WEIGHTED-FVS. Later, we solve the general problem, using the algorithm of the special case.

3.1 RESTRICTED BGVD

In this part, we solve the following special case of BGVD in FPT time.

RESTRICTED BGVD	**Parameter:** k
Input: A connected undirected graph G, which is $\{D_4, C_4\}$-free, and a positive integer k.	
Question: Does there exist a set S such that $G \setminus S$ is a block graph?	

Let G be the input graph. Let $\mathcal{C}$ be the set of maximal cliques in G. We start with the following simple observation about graphs without C_4 and D_4.

Lemma 1. *Let G be a graph that does not contain C_4 and D_4 as an induced subgraph then (a) any two maximal cliques intersect on at most one vertex and (b) the number of maximal cliques in G is at most n^2.*

Proof. Let C_1 and C_2 be two maximal cliques in $\mathcal{C}$. Since G is D_4-free, $V(C_1) \cap V(C_2)$ can have at most one vertex. Thus, each edge of G belongs to exactly one maximal clique. This gives a bound of n^2 on the number of maximal cliques. □

We construct an auxiliary weighted bipartite graph $\hat{G}$ in the following way: $\hat{G}$ is a bipartite graph with vertex set bipartition $V(G) \cup V_{\mathcal{C}}$, where $V_{\mathcal{C}}$ is the set where we add a vertex v_C corresponding to each $C \in \mathcal{C}$. Note that there is a bijective correspondence between the vertices of $V_{\mathcal{C}}$ and the maximal cliques in $\mathcal{C}$. A vertex v of a clique C is called *external* if it is part of at least two maximal cliques in $\mathcal{C}$. We add an edge between a vertex $v \in V(G)$ and a vertex $v_C \in V_{\mathcal{C}}$ in $E(\hat{G})$ if and only if v is an external vertex of the clique $C \in \mathcal{C}$.

Lemma 2. *Let G be a graph without induced C_4 and D_4 and $S \subseteq V(G)$. Then S is block vertex deletion set of G if and only if $\hat{G} \setminus S$ is acyclic.*

Proof. First, let S be a block vertex deletion set solution for G. Suppose that $\hat{G} \setminus S$ has a cycle C. Notice that C cannot be a C_4, as this corresponds to two maximal cliques that share 2 vertices. Thus, C is an even cycle of length at least 6. Suppose C has length 6. This corresponds to maximal cliques C_1, C_2, C_3 such that $u = C_1 \cap C_2$, $v = C_2 \cap C_3$ and $w = C_1 \cap C_3$. Since C_1, C_2, C_3 are distinct maximal cliques, at least one of them must have a vertex other than u, v or w. Without loss of generality, let C_1 have a vertex $x \notin \{u, v, w\}$. Then, the set $\{x, u, v, w\}$ forms a D_4 in G. However, this is not possible, as G did not have a D_4 to start with. Hence, C must be an even cycle of length at least 8. However, this corresponds to a set of maximal cliques and external vertices, such that the external vertices form an induced cycle of length at least four. This contradicts that S was a block vertex deletion set for G. Thus, $\hat{G} \setminus S$ must be acyclic.

On the other hand, let $\hat{G} \setminus S$ be acyclic. Suppose $G \setminus S$ has an induced cycle C, of length at least four. As C is an induced cycle of length at least four, no two edges of C can belong to the same maximal clique. For an edge (u, v) of C, let $C_{(u,v)}$ be the maximal clique containing it. Also, let $c_{(u,v)}$ be the corresponding vertex in $\hat{G}$. We replace the edge (u, v) in C by two edges $(u, c_{(u,v)})$ and $(v, c_{(u,v)})$. In this way, We obtain a cycle C' of $\hat{G} \setminus S$, which is a contradiction. Thus, S must be a block vertex deletion set for G. $\square$

If the input graph G is without induced C_4 and D_4 then Lemma 2 tells us that to find block vertex deletion set of G of size at most k one can check whether there is a feedback vertex set of size at most k for $\hat{G}$ contained in $V(G)$. To enforce that we find feedback vertex set for $\hat{G}$ completely contained in $V(G)$ we solve an appropriate instance of Weighted-FVS. In particular we give the weight function $w : V(\hat{G}) \rightarrow \mathbb{N}$ as follows. For $v \in V(G)$, $w(v) = 1$ and for $v_C \in V_{\mathcal{C}}$, $w(v_C) = n^4$. Clearly, $V(G)$ is a feedback vertex set of $\hat{G}$ and thus the weight of a minimum sized feedback vertex set of $\hat{G}$ is at most n. This implies that running an algorithm for Weighted-FVS on an instance $(\hat{G}, w, k)$ either returns a feedback vertex set contained inside $V(G)$ or returns that the given instance is a No instance.

Theorem 5. Restricted BGVD *can be solved in* $\mathcal{O}^\star(3.618^k)$.

Proof. Given an instance (G, k) of Restricted BGVD. We apply the Weighted-FVS on the instance $(\hat{G}, w, k)$, where $\hat{G}$ is obtained as described above. Let S be the weighted-fvs of size at most k in $\hat{G}$ returned by Weighted-FVS (of course if there exists one). By the discussion above we know that if Weighted-FVS does not return that the given instance is a No instance then $S \subseteq V(G)$. If it returns that the given instance is a No instance then we return the same. Else, assume that S is non-empty. Now we check whether $w(S)$ is at most k or not. Since every vertex in $V(G)$ has been assigned weight one we have that $w(S) = |S|$ and thus if $w(S) \leq k$ then we return S as block vertex deletion set of G. In the case when $w(S) > k$ we return that the given instance is a No instance for Restricted BGVD. Correctness of these steps are guaranteed by Lemma 2. The running time of the algorithm is dominated by the running time of Weighted-FVS and thus it is $\mathcal{O}^\star(3.618^k)$. This completes the proof. $\square$

3.2 Block Graph Vertex Deletion

We are now ready to describe an FPT algorithm for BGVD, and hence prove Theorem 1. We design the algorithm for the general case with the help of the algorithm for RESTRICTED BGVD.

Proof (of Theorem 1). Let O be a D_4 or C_4 present in the input graph G. For any potential solution S, at least one of the vertices of O must belong to S. Therefore, we branch on the choice of these vertices, and for every vertex $v \in O$, we recursively apply the algorithm to solve BGVD instance $(G \setminus \{v\}, k-1)$. If one of these branches returns a solution X, then clearly $X \cup \{v\}$ is a block vertex deletion set of size at most k for G. Else, we return that the given instance is a NO instance. On the other hand, if G is $\{D_4, C_4\}$-free, then we do not make any further recursive calls. Instead, we run the algorithm for RESTRICTED BGVD on G and return the output of the algorithm. Thus, the running time of this algorithm is upper bounded by $\mathcal{O}^*(4^k)$. □

4 An Approximation Algorithm for BGVD

In this section, we present a simple approximation algorithm $\mathcal{A}_1$ for BGVD. Given a graph G, we give a block vertex deletion set S of size at most $4 \cdot \mathsf{OPT}$, where OPT is the size of a minimum sized block vertex deletion set for G.

Proof (of Theorem 2). Let G be the given instance of BGVD and OPT be the size of a minimum sized block vertex deletion set for G and S_{OPT} be a minimum sized block vertex deletion set for G.

Let $\mathcal{S}$ be a maximal family of D_4 and C_4 such that any two members of $\mathcal{S}$ are pairwise disjoint. One can easily construct such a family $\mathcal{S}$ greedily in polynomial time. Let S_1 be the set of vertices contained in any obstruction in $\mathcal{S}$. That is, $S_1 = \bigcup_{O \in \mathcal{S}} O$. Since any block vertex deletion set must contain a vertex from each obstruction in $\mathcal{S}$ and any two members of $\mathcal{S}$ are pairwise disjoint, we have that $|S_{\mathsf{OPT}} \cap S_1| \geq |\mathcal{S}|$.

Let $G' = G \setminus S_1$. Observe that G' does not contain either D_4 or C_4 as an induced subgraph. Now we construct $\hat{G}'$, as described in Sect. 3.1. We apply the factor two approximation algorithm $\mathcal{A}$ given in [1] on the instance $(\hat{G}', w)$. This returns an fvs S_2 of $\hat{G}'$ such that $w(S_2)$ is at most twice the weight of a minimum weight feedback vertex set. By out construction $S_2 \subseteq V(G')$. Lemma 2 implies that S_2 is a factor two approximation for BGVD on G'. We return the set $S = S_1 \cup S_2$ as our solution. Since $S_{\mathsf{OPT}} \setminus S_1$ is also an optimum solution for G' we have that $|S_2| \leq 2|S_{\mathsf{OPT}} \setminus S_1|$.

It is evident that S is block vertex deletion set of G. To conclude the proof of the theorem we will show that $|S| \leq 4\mathsf{OPT}$. Towards this observe that

$$\begin{aligned} |S| = |S_1| + |S_2| &\leq 4|\mathcal{S}| + 2|S_{\mathsf{OPT}} \setminus S_1| \\ &\leq 4|S_{\mathsf{OPT}} \cap S_1| + 2|S_{\mathsf{OPT}} \setminus S_1| \\ &\leq 4|S_{\mathsf{OPT}}| = 4\mathsf{OPT}. \end{aligned}$$

This completes the proof. □

5 Improved Kernel for Block Graph Vertex Deletion

In this section, we give a kernel of $\mathcal{O}(k^4)$ vertices for BGVD. Let (G, k) be an instance of the BGVD problem. We start with some of the known reduction rules from [10].

Reduction Rule BGVD 1. *If G has a component H, where H is a block graph, then remove H from G.*

Reduction Rule BGVD 2. *If there is a vertex $v \in V(G)$, such that $G \setminus \{v\}$ has a component H, where $G[\{v\} \cup V(H)]$ is a connected block graph then, remove H from G.*

Reduction Rule BGVD 3. *Let $S \subseteq V(G)$, where each $u, v \in S$ are true-twins in G. If $|S| > k + 1$, then remove all the vertices from S except $k + 1$ vertices.*

Reduction Rule BGVD 4. *Let t_1, t_2, t_3, t_4 be an induced path in G. For $i \in \{1, 2, 3\}$, let $S_i \subseteq V(G) \setminus \{t_1, t_2, t_3, t_4\}$ be a clique in G such that the following holds.*

- *For $i \in \{1, 2, 3\}$, $v \in S_i$, $N_G(v) \setminus S_i = \{t_i, t_{i+1}\}$, and*
- *For $i \in \{2, 3\}$, $N_G(t_i) = \{t_{i-1}, t_{i+1}\} \cup S_{i-1} \cup S_i$.*

Remove S_2 from G and contract the edge (t_2, t_3).

Proposition 1 (Proposition 3.1 [10]). *Let G be a graph and k be a positive integer. For a vertex $v \in V(G)$, in $\mathcal{O}(kn^3)$ time, we can find one of the following.*

i. *$k + 1$ pairwise vertex disjoint obstructions,*
ii. *$k + 1$ obstructions whose pairwise intersection is exactly v,*
iii. *$S'_v \subseteq V(G)$, such that $|S'_v| \leq 7k$ and $G \setminus S'_v$ has no block graph obstruction containing v.*

Reduction Rule BGVD 5. *Let $v \in V(G)$ and $G' = G \setminus \{v\}$. We remove the edges between $N_G(v)$ from G', i.e. $E(G') = E(G') \setminus \{(u, w) | u, w \in N_G(v)\}$. In G' if there are at least $2k + 1$ vertex-disjoint $N_G(v)$-paths in G' then we do one of the following.*

- *If G contains $k + 1$ vertex disjoint obstructions, then return that the graph is a no-instance.*
- *Otherwise, delete v from G and decrease k by 1.*

The Reduction rules BGVD 1 to BGVD 5 are safe and can be applied in polynomial time [10]. For sake of clarity we denote the reduced instance at each step by (G, k). We always apply the lowest numbered Reduction Rule, in the order that they have been stated, that is applicable at any point of time. For the rest of the discussion, we assume that Reduction rules BGVD 1 to BGVD 5 are not applicable.

For a vertex $v \in V(G)$, by Proposition 1, we may find $k+1$ pairwise vertex-disjoint obstructions, and we can safely conclude that the graph is a No instance. Secondly, if we find $k+1$ obstructions whose pairwise intersection is exactly v then the Reduction rule BGVD 5 will be applicable. Thus, we assume that for each vertex $v \in V(G)$, the third condition of Proposition 1 holds. In other words, we have a set S'_v of size at most $7k$, such that $G \setminus S'_v$ does not contain any obstruction passing through v. In fact, for each $v \in V(G)$, we can find a block vertex deletion set $S_v \subseteq V(G) \setminus \{v\}$ of bounded size.

Observation 1 [$\star$]. *For every vertex $v \in V(G)$, we can find in $n^{\mathcal{O}(1)}$ time, a set $S_v \subseteq V(G) \setminus \{v\}$ such that $|S_v| \leq 11k$ and $G \setminus S_v$ is a block graph.*

For a vertex $v \in V(G)$, *component degree* of v is the number of connected components in $\mathcal{C}$, where $\mathcal{C}$ is the set of connected components in $G \setminus (S_v \cup \{v\})$ that have a vertex adjacent to v. We give a reduction rule that bounds the *component degree* of a vertex $v \in V(G)$, using *Expansion Lemma* [15].

A *q-star*, $q \geq 1$, is a graph with $q+1$ vertices, one vertex of degree q and all other vertices of degree 1. Let $\mathcal{B}$ be a bipartite graph with the vertex bipartition as (X, Y). A set of edges $M \subseteq E(\mathcal{B})$ is called a *q-expansion* of X into Y if (i) every vertex of X is incident with exactly q edges of M and (ii) M saturates exactly $q|X|$ vertices in Y, i.e. edges in M are adjacent to exactly $q|X|$ vertices in Y.

Lemma 3 (Expansion Lemma). *Let q be a positive integer and $\mathcal{B}$ be a bipartite graph with vertex bipartition (X, Y) such that $|Y| \geq q|X|$ and there are no isolated vertices in Y. Then, there exist nonempty vertex sets $X' \subseteq X$ and $Y' \subseteq Y$ such that:*

1. *X' has a q-expansion into Y' and*
2. *no vertex in Y' has a neighbour outside X', i.e. $N(Y') \subseteq X'$.*

Furthermore, the sets X' and Y' can be found in polynomial time.

See [4] for the version of the Lemma 3 stated above. For a vertex $v \in V(G)$, let $\mathcal{C}_v$ be the set of connected components in $G \setminus (S_v \cup \{v\})$ that have a vertex adjacent to v. Consider a connected component $C \in \mathcal{C}_v$, such that no vertex $u \in V(C)$ is adjacent to any vertex in S_v. But then, $G \setminus \{v\}$ has a component which is a block graph (namely, the connected component C) therefore, Reduction rule BGVD 2 is applicable, a contradiction to the assumption that none of the previous Reduction rules are applicable. Therefore, for each $C \in \mathcal{C}$ there is a vertex $u \in V(C)$ and $s \in S_v$, such that $(u, s) \in E(G)$. Let $\mathcal{D}$ be a vertex set, with a vertex d corresponding to each component $D \in \mathcal{C}$. Consider the bipartite graph $\mathcal{B}_v$ with the vertex set bipartitioned as $(\mathcal{D}, S_v)$. There is an edge between $d \in \mathcal{D}$ and $s \in S_v$ if and only if the component D corresponding to which the vertex d was added to $\mathcal{D}$ has a vertex u_d such that $(u_d, s) \in E(G)$.

Reduction Rule BGVD 6. *For a vertex $v \in V(G)$ if $|\mathcal{C}_v| > 33k$, then we do the following.*

- *Let $\mathcal{D}' \subseteq \mathcal{D}$ and $S \subseteq S_v$ be the sets obtained after applying Lemma 3 with $q = 3$, $X = S_v$ and $Y = \mathcal{D}$;*
- *For each $d \in \mathcal{D}'$, let the component corresponding to d be $D \in \mathcal{C}_v$. Delete all the edges between (u, v), where $u \in V(D)$;*
- *For each $s \in S$, add two vertex disjoint paths between v and s.*

Safeness of the Reduction rule BGVD 6 follows from the safeness of Reduction rule 6 in [10].

5.1 Bounding the Number of Blocks in $G \setminus A$

Using the approximation algorithm for BGVD we compute an approximate solution A of size at most $4k$. Of course if $|A| > 4k$ then we can immediately return that G is a No instance. First, we bound the number of leaf blocks in $G \setminus A$, when none of the Reduction rules apply. Note that $G \setminus A$ is a block graph, since A is an approximate solution to BGVD. For $v \in A$, let S'_v be the set obtained from Proposition 1 and S_v be the set obtained from Observation 1. Let $\mathcal{C}_v$ be the set of connected components in $G \setminus (S_v \cup \{v\})$ which have a vertex adjacent to v. All the connected components in $G \setminus A$, which do not have a vertex that is adjacent to v, must be adjacent to some $v' \in A$. Otherwise, Reduction rule BGVD 1 will be applicable. Also, all the leaf blocks in $G \setminus A$ must have an internal vertex that is adjacent to some vertex in A, since the Reduction rules BGVD 1 and BGVD 2 are not applicable. The number of leaf blocks, in $G \setminus A$, whose set of internal vertices have a non-empty intersection with S'_v, is at most $7k$. Therefore, it is enough to count, for each $v \in A$, the number of leaf blocks in $\mathcal{C}_v$. In the Observation 2, we give a bound on the number of leaf blocks in $G \setminus A$, not containing any vertex from S'_v.

Observation 2 [$\star$]**.** *For $v \in A$, the number of leaf blocks in $G \setminus A$ not containing any vertex from S'_v is at most the number of leaf blocks in $G \setminus (S_v \cup \{v\})$.*

Therefore, for each $v \in A$ we count those leaf blocks in $\mathcal{C}_v$ which do not contain any vertex from S'_v.

Lemma 4. *Consider a vertex $v \in V(G)$ and its corresponding set S_v. Let $\mathcal{C}$ be the set of connected components in $G \setminus (S_v \cup \{v\})$. For each $C \in \mathcal{C}$, there is a block $\tilde{B}$ in C, such that $N_C(v) \subseteq V(\tilde{B})$.*

Proof. Let $\mathcal{C}$ be the set of connected components of $G \setminus (S_v \cup \{v\})$, $v \in V(G)$. By definition of S_v, for each $C \in \mathcal{C}$, $C \cup \{v\}$ is a block graph.

If for some $C \in \mathcal{C}$, $N_C(v) = \emptyset$, then the condition is trivially satisfied for that connected component C. Let $C \in \mathcal{C}$ be a connected component such that $N_C(v) \neq \emptyset$. Let t be a vertex in $N_B(v)$, where B is a block in C. Let B' be a block in C, where $B' \neq B$ and B' has a vertex $t' \in V(B') \setminus V(B)$ that is adjacent to v. Note that B, B' are in the same connected component C. Let P be the shortest path from t to t'.

We first argue for the case when $(t, t') \notin E(G)$. Therefore, the path P has at least 2 edges. We prove that we can find an obstruction, by induction on the length of the path (number of edges). If length of path P is 2, say $P = t, u, t'$. If $(u, v) \in E(G)$, then $\{t, t', u, v\}$ forms an induced D_4, otherwise they form an induced C_4, contradicting that $C \cup \{v\}$ is a block graph.

Let us assume that we can find an obstruction if the path length is l. We now prove it for paths of length $l+1$. Let $P = t, x_1, x_2, \ldots, x_{l-1}, t'$ and y be the first vertex other than t in P such that $(y, v) \in E(G)$. If $y = t'$, then P along with v forms an induced cycle of length at least 5, contradicting that $C \cup \{v\}$ is a block graph. If $y = x_1$, then $\{t, x_1, x_2, v\}$ either forms a D_4, the case when $(x_2, v) \in E(G)$, or $\hat{P} = x_1, x_2, \ldots, t'$ is a path of shorter length with at least 2 edges and by induction hypothesis has an obstruction along with v. Otherwise, $P' = t, x_1, \ldots, y$ is a path of length less than l, with at least 2 edges, such that $(y, t) \in E(G)$. Therefore, by induction hypothesis there is an obstruction along with the vertex v, contradicting that $C \cup \{v\}$ is a block graph.

From the above arguments it follows that if v has a neighbour t in block B in C, then v cannot have a neighbour t' in block B', if the shortest path between t, t' has at least 2 edges.

If $(t, t') \in E(G)$, then t, t' are contained in some block $\hat{B}$. If v is adjacent to any other vertex u not in $V(\hat{B})$ then at most one of (t, u) or (t', u) can be an edge in G, since t, t' and u are in different blocks. If there is an edge, say (t, u), then t, t', u, v forms an induced D_4, contradicting that $C \cup \{v\}$ is a block graph. Otherwise, there is a path with at least two edges between u and t. Therefore, by the previous arguments we can find an obstruction along with the vertex v. Therefore, $N_C(v) \subseteq V(\hat{B})$ when $(t, t') \in E(G)$.

Hence, it follows that there is a block $\tilde{B}$ in C such that $N_C(v) \subseteq V(\tilde{B})$. □

This leads us to the following Lemma.

Lemma 5 [$\star$]. *For every $v \in A$, the number of leaf blocks in $\mathcal{C}_v$ is $\mathcal{O}(k)$.*

Observe that in $G \setminus A$, a vertex $v \in A$ can be adjacent to at most $\mathcal{O}(k)$ leaf blocks by Observation 2 and Lemma 5. Also, for a leaf block B in $G \setminus A$, there must be an internal vertex $b \in V(B)$, such that b is adjacent to some vertex in S_v, since the Reduction rule BGVD 2 is not applicable. Therefore, the number of leaf blocks in $G \setminus A$ is $\mathcal{O}(k^2)$.

Lemma 6 [$\star$]. *The number of blocks B in $G \setminus A$ such that the vertex set of B intersects with the vertex set of at least three other block in $G \setminus A$ is $\mathcal{O}(k^2)$.*

Let $\mathcal{L}$ be the of leaf blocks in $G \setminus A$ and $\mathcal{T}$ be the set of blocks in $G \setminus A$ such that each block in $\mathcal{T}$ intersects with at least three other blocks in $G \setminus A$. By Lemmas 5 and 6, we have that $|\mathcal{L}| = \mathcal{O}(k^2)$ and $|\mathcal{T}| = \mathcal{O}(k^2)$.

Let B be a block in $G^* = G \setminus (S_v \cup \{v\})$ such that the vertex set of B has exactly two cut vertices, and intersects with exactly two blocks of G^*. Furthermore, the vertex set of B has an empty intersection with leaf blocks of G^* and those blocks in G^* which vertex set intersects with at least three other blocks of G^*. Also, B has a vertex that is neighbor to v. Such blocks are called *nice degree two blocks* of v. If a block satisfies the above conditions for some vertex $w \in A$, the block is called a *nice degree two block*. We denote the set of nice degree two blocks by $\mathcal{T}_1$.

Lemma 7 [$\star$]. *Let $G^* = G \setminus (S_v \cup \{v\})$. Then G^* has at most $\mathcal{O}(k)$ nice degree two blocks of v.*

What remains is to bound the number of blocks which have exactly two cut vertices and are not nice degree two blocks.

Lemma 8 [$\star$]. *The number of blocks in $G \setminus A$ with exactly two cut vertices is $\mathcal{O}(k^2)$.*

Now, we have a bound on the total number of blocks in $G \setminus A$.

Lemma 9. *Consider a graph G, a positive integer k and an approximate block vertex deletion set set A of size $\mathcal{O}(k)$. If none of the Reduction rules* BGVD *1 to* BGVD *6 is applicable then the number of blocks in $G \setminus A$ is bounded by $\mathcal{O}(k^2)$.*

Proof. Follows from Lemmas 5, 6 and 8. □

5.2 Bounding the Number of Internal Vertices in a Maximal Clique of the Block Graph

We start by bounding the number of internal vertices in a maximal 2-connected component of $G \setminus A$. Consider a block B in $G \setminus A$. We partition the internal vertices $V_I(B)$ of block B into three sets $\mathcal{B}, \mathcal{R}$ and $\mathcal{I}$ depending on the neighborhood of A in block B. We also partition the vertices in A depending on the number of vertices they are adjacent to in B. In Lemma 10 we show that the number of internal vertices in a block B of $G \setminus A$ is upper bounded by $\mathcal{O}(k^2)$. We do so by partitioning the vertices into different sets and bounding each of these sets separately.

Lemma 10 [$\star$]. *Let (G, k) be an instance to* BGVD *and let A be an approximate block vertex deletion set of G of size $\mathcal{O}(k)$. If none of the Reduction rules* BGVD *1 to* BGVD *6 is applicable then the number of internal vertices in a block B of $G \setminus A$ is bounded by $\mathcal{O}(k^2)$.*

We wrap up our arguments to show a $\mathcal{O}(k^4)$ sized vertex kernel for BGVD, and hence prove Theorem 3.

Proof (of Theorem 3). Let (G, k) be an instance to BGVD and let A be an approximate block vertex deletion set of G of size $\mathcal{O}(k)$. Also, assume that none of the Reduction rules BGVD 1 to BGVD 6 are applicable. By Theorem 9, the number of blocks in $G \setminus A$ is bounded by $\mathcal{O}(k^2)$. By Lemma 10 the number of internal vertices in a block of $G \setminus A$ is bounded by $\mathcal{O}(k^2)$. Also note that the number of cut-vertices in $G \setminus A$ is bounded by the number of blocks in $G \setminus A$, i.e. $\mathcal{O}(k^2)$. The number of vertices in $G \setminus A$ is sum of the internal vertices in $G \setminus A$ and the number of cut vertices in $G \setminus A$. Therefore, $|V(G)| = |V(G \setminus A)| + |A| = (\mathcal{O}(k^2) \cdot \mathcal{O}(k^2) + \mathcal{O}(k^2)) + \mathcal{O}(k) = \mathcal{O}(k^4)$. □

References

1. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM J. Discret. Math. **12**(3), 289–297 (1999)
2. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: A Survey. Society for Industrial and Applied Mathematics, Philadelphia (1999)
3. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. J. Comput. Syst. Sci. **74**(7), 1188–1198 (2008)
4. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Switzerland (2015)
5. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Heidelberg (2012)
6. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, New York (2006)
7. Fomin, F.V., Lokshtanov, D., Misra, N., Saurabh, S., F-deletion, P.: Approximation, kernelization and optimal FPT algorithms. In: FOCS (2012)
8. Fujito, T.: A unified approximation algorithm for node-deletion problems. Discrete Appl. Math. **86**, 213–231 (1998)
9. Howorka, E.: A characterization of ptolemaic graphs. J. Graph Theor. **5**(3), 323–331 (1981)
10. Kim, E.J., Kwon, O.: A polynomial kernel for block graph vertex deletion. CoRR, abs/1506.08477 (2015)
11. Kociumaka, T., Pilipczuk, M.: Faster deterministic feedback vertex set. Inf. Process. Lett. **114**(10), 556–560 (2014)
12. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980)
13. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. J. ACM **41**, 960–981 (1994)
14. Marx, D., O'Sullivan, B., Razgon, I.: Finding small separators in linear time via treewidth reduction. ACM Trans. Algorithms **9**(4), 30 (2013)
15. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Trans. Algorithms **6**(2), 32: 1–32: 8 (2010)
16. Tong, P., Lawler, E.L., Vazirani, V.V.: Solving the weighted parity problem for gammoids by reduction to graphic matching. Technical report UCB/CSD-82-103, EECS Department, University of California, Berkeley, April 1982