A faster FPT algorithm and a smaller Kernel for Block Graph Vertex Deletion[☆]

Akanksha Agrawal[a,*], Sudeshna Kolay[b], Daniel Lokshtanov[a], Saket Saurabh[a,c]

[a] *University of Bergen, Bergen, Norway.*
[b] *Technische Universiteit Eindhoven, Eindhoven, Netherlands.*
[c] *The Institute of Mathematical Sciences, HBNI, Chennai, India*

Abstract

A graph is a block graph if each maximal 2-connected component of it is a clique. In this paper, we study the problem Block Graph Vertex Deletion (BGVD) from the perspective of fixed parameter tractable (FPT) and kernelization algorithms. An input to BGVD consists of a graph G and an integer k, and the objective is to decide if there is $S \subseteq V(G)$ such that $G[V(G) \setminus S]$ is a block graph and $|S| \leq k$. In this paper, we give an FPT algorithm running time $4^k n^{\mathcal{O}(1)}$ and a kernel of size $\mathcal{O}(k^4)$ for BGVD. These results improve over the previous best known FPT algorithm and kernel for the problem. Our results are based on a connection between BGVD and the classical Feedback Vertex Set problem. To achieve our results we design an algorithm for Weighted Feedback Vertex Set running in time $3.618^k n^{\mathcal{O}(1)}$, the previous best algorithm for which runs in time $5^k n^{\mathcal{O}(1)}$.

Keywords: Block Graph Vertex Deletion, Weighted Feedback Vertex Set, Parameterized Complexity, FPT Algorithms, Kernels

1. Introduction

Deleting 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 some graph 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

[☆]A preliminary version of this manuscript appeared in the 12th Latin American Theoretical Informatics Symposium (LATIN 2016).
The research leading to these results received funding from the BeHard grant under the recruitment programme of the of Bergen Research Foundation and the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreements no. 306992 (PARAPPROX).

[*]Corresponding author.
Email addresses: `akanksha.agrawal@uib.no` (Akanksha Agrawal), `s.kolay@tue.nl` (Sudeshna Kolay), `daniello@ii.uib.no` (Daniel Lokshtanov), `saket@imsc.res.in` (Saket Saurabh)

graph belongs to the family $\mathcal{F}$. Lewis and Yannakakis [1] 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 [2, 3, 4, 5]. These paradigms among others include applying restrictions on inputs, approximation algorithms and parameterized algorithms. The focus of this paper is to study one such problem from the view point of parameterized algorithms.

Given a family $\mathcal{F}$, a typical parameterized vertex deletion problem takes as an input a graph G and an integer k, and the objective is to decide whether or not there is $S \subseteq V(G)$ of size at most k such that $G - S \in \mathcal{F}$. In the Parameterized Complexity paradigm one of the main objectives 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 $f(\cdot)$ is an arbitrary (computable) function depending only on k. Such an algorithm is called an FPT algorithm, and such a running time is called FPT running time. We also desire to 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 $h(k)$. If the function $h(\cdot)$ is polynomial in k, then we say that the problem admits a polynomial kernel. The formal definitions are given in Section 2. For more details on Parameterized Complexity we refer to the books of Downey and Fellows [6, 7], Flum and Grohe [8], Niedermeier [9], and Cygan et al. [10].

A large part of research in paramaterized vertex deletion problems has centred around $\mathcal{F}$, that can be defined by finite excluded minor characterization, and in particular, when $\mathcal{F}$ has a planar graph [2, 11]. These include problem of deleting k vertices to get a graph of bounded (fixed) treewidth [2, 11]. One of the main reason to study these problems is that several NP-hard problems become polynomial time solvable on graphs of bounded treewidth. However, there are several other notions similar to treewidth where several NP-hard problems become polynomial time solvable; these include cliquewidth, rankwidth, and linear rankwidth. Recently, Kanté et al. [12] studied the problem of deleting k vertices to get a graph of linear rankwidth one and in an other study Kim and Kwon [13] studied deleting k vertices to get a *block graph*.

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 $K_4 - e$ [14, 15]. The class of block graphs is the intersection of the chordal and distance-hereditary graphs [15].

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 $G - S$ is a block graph. The NP-hardness of BGVD follows from [1].

Block Graph Vertex Deletion (BGVD)	**Parameter:** k

Input: A graph G and an integer k.
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a block graph?

Kim and Kwon [16, 13] gave an FPT algorithm with running time $\mathcal{O}(10^k |V(G)|^{\mathcal{O}(1)})$ for the problem. Also, they designed a kernel with $\mathcal{O}(k^9)$ vertices [16], which they improved to a kernel with $\mathcal{O}(k^6)$ vertices [13]. We improve both these results via a novel connection to the 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 two main results are:

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

Theorem 2. BGVD *has a kernel of size* $\mathcal{O}(k^4)$.

Our two theorems improve both the results in [13]. That is, the running time of our FPT algorithm improves over the previous best algorithm for the problem which runs in time $10^k n^{\mathcal{O}(1)}$ and the size of our kernel reduces over the previously known kernel which was 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 input graph does not contain induced cycles on four vertices 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 [13]. 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 a graph G and an integer k, the objective is to decide whether or not there is $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest. Thus, S is a vertex subset that intersects with every cycle in G. In the Parameterized Complexity setting, Feedback Vertex Set parameterized by k, has an FPT algorithm. The best known FPT algorithm runs in time $\mathcal{O}(3.618^k n^{\mathcal{O}(1)})$ [10, 17]. The problem also admits a kernel on $\mathcal{O}(k^2)$ vertices [18]. Another variant of Feedback Vertex Set that has been studied in Parameterized Complexity is Weighted Feedback Vertex Set (Weighted-FVS), where each vertex in the graph has some rational number as its weight.

WEIGHTED-FVS	**Parameter:** k
Input: A graph G, a weight function $w : V(G) \to \mathbb{Q}$, and an integer k.	
Output: A set $S \subseteq V(G)$ of size at most k of minimum possible weight such that $G - S$ is a forest.	

WEIGHTED-FVS is known to be in FPT with an algorithm of running time $5^k n^{\mathcal{O}(1)}$ [19]. We obtain a faster FPT algorithm for WEIGHTED-FVS. This algorithm uses, as a subroutine, the algorithm for solving WEIGHTED-MATROID PARITY, which is solvable in polynomial time on linear matroids [20]. In fact, this algorithm is very similar to the algorithm for FEEDBACK VERTEX SET given in [10]. Thus, our final new result is the following theorem.

Theorem 3. WEIGHTED-FVS *has an* FPT *algorithm which runs in time* $\mathcal{O}(3.618^k|V(G)|^{\mathcal{O}(1)})$.

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 denote the set of natural numbers by $\mathbb{N}$. For $k \in \mathbb{N}$, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. To describe the running times of our algorithms, we will use the $\mathcal{O}^\star$ notation. 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.

Graphs. We use standard terminology from the book of Diestel [21] for the graph related terminologies which are not explicitly defined here. For a graph G, by $V(G)$ and $E(G)$ we denote the vertex and edge sets of G, respectively. All graphs that we consider are finite graphs, possibly having loops and multi-edges. For $W \subseteq V(G)$, 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 in G whose both endpoints are in W. For $W \subseteq V(G)$, by $G - W$ 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 $v \in V(G)$ by $d_G(v)$. A vertex $v \in V(G)$ is called as a *cut vertex* if the number of connected components in $G - \{v\}$ 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 $N_G(v) = \{u \mid (v, u) \in E(G)\}$. 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 $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. A *path* $P = (v_1, v_2, \cdots, v_\ell)$ in G is a subgraph of G with vertex set as $\{v_1, v_2, \cdots, v_\ell\}$ and edge set as $\{(v_i, v_{i+1}) \mid i \in [l - 1]\}$. Furthermore, we call such a path as a path between v_1 and v_ℓ, where v_1 and v_ℓ are end vertices and $v_2, \ldots, v_{\ell-1}$ are internal vertices. The length of a path is the number of edges in it. For $A \subseteq V(G)$, 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 $V(G) \setminus A$.

Consider $u, v \in V(G)$ such that $u \neq v$ and neither u nor v has a self loop. By contracting the edge $(u, v) \in E(G)$ we mean the following operation. We create a new graph G', where $V(G') = (V(G) \setminus \{u, v\}) \cup \{uv^*\}$ and $E(G') = E(G[V(G) \setminus \{u, v\}]) \cup \{(uv^*, w) \mid w \in (N(u) \cup N(v)) \setminus \{u, v\}\}$. Note that there is bijection $f : E(G) \setminus \{(u, v)\} \to E(G')$ as follows. For $(x, y) \in E(G)$ $f((x, y)) = (x, y)$ if $x, y \notin \{u, v\}$. For all other edges, one of x, y is same as u, v. Note that the edge (x, y) is not same as the edge (u, v), and $u \neq v$ implies $x \neq y$. The only case left is when exactly one of x or y is same as one of u, v say, $x = u$ then, $f((x, y)) = (uv^*, y)$. Analogously, we can find $f((x, y))$ for the remaining cases. Slightly abusing the notation, for an edge $e \in E(G) \setminus \{(u, v)\}$ we will refer to $f(e) \in E(G')$ by e.

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

A *feedback vertex set* is a set $S \subseteq V(G)$ such that $G - S$ is a forest. A minimum weighted-fvs of a weighted graph G is a set $X \subseteq V(G)$ such that $G - X$ 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 set $X \subseteq V(G)$ such that $|X| \leq k$, $G - X$ 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 $G - S$ is a block graph.

A family $\mathcal{F}$ of sets over a universe U is called a matroid if it satisfies the following properties.

- $\emptyset \in \mathcal{F}$;
- if $A \in \mathcal{F}$ and $B \subseteq A$, then $B \in \mathcal{F}$;
- if $A, B \in \mathcal{F}$ and $|A| < |B|$, then there exists $b \in B \setminus A$ such that $A \cup \{b\} \in \mathcal{F}$.

For a matroid $(U, \mathcal{F})$, the elements of U are called edges and sets in $\mathcal{F}$ are called independent sets. For an undirected graph G, a graphic matroid $\mathcal{M}_G$ is a matroid with $U = E(G)$, and a set $S \subseteq E(G)$ is an independent set in $\mathcal{M}_G$ if the graph $G' = (V(G), S)$ is acyclic. Observe that $\mathcal{M}_G$ satisfies all the properties required for it to be a matroid.

A graph G is called a block graph if every maximal 2-connected component in G is a clique. A maximal 2-connected subgraph in G is called a *block*. Another characterization of a block graph is a graph that has no induced cycles of length at least 4 and no induced $K_4 - e$ [14]. Here, $K_4 - e$ is a complete graph on 4 vertices with one edge removed. For a graph G, let V_c denote the set of cut vertices in 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 $v \in V(B)$. Note that for a block graph G, the bipartite graph F is a forest [21]. The bipartite graph F is called as the block forest of H. 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.

Parameterized Complexity. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$, where Γ is a finite alphabet set. An instance of a parameterized problem is a tuple (x, k), where x is a classical problem instance and k is an integer, which is called the parameter. A central notion in Parameterized Complexity is *fixed-parameter tractability (FPT)* which means, for a parameterized problem Π, and an instance (x, k) of Π, decidability of whether or not $(x, k) \in \Pi$ in time $f(k) \cdot p(|x|)$. Here, $f(\cdot)$ is an arbitrary (computable) function of k and p is a polynomial in the input size.

Kernelization. A kernelization algorithm for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that, given (x, k) of Π, outputs, in time polynomial in $|x| + k$, an instance (x', k') of Π such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where $g(\cdot)$ is some computable function. The output instance (x', k') is called the kernel, and the function $g(\cdot)$ is referred to as the size of the kernel. If $g(k) = k^{\mathcal{O}(1)}$ (resp. $g(k) = \mathcal{O}(k)$) then we say that Π admits a polynomial (resp. linear) kernel.

3. Improved Algorithm for Weighted Feedback Vertex Set

In this section, we give an improved algorithm for the Weighted-FVS. We use the method of iterative compression together with branching and reduce Weighted-FVS to the problem of Weighted-Matroid Parity, which can be solved in polynomial time. The algorithm we give is similar to the algorithm for Feedback Vertex Set in [10, 17]. We give an algorithm only for the disjoint variant of the problem.

Observation 1 ([10]). *The existence of an algorithm for the disjoint variant of* Weighted-FVS *with running time $c^k \cdot n^{\mathcal{O}(1)}$, for a constant c, implies that* Weighted-FVS *can be solved in time $c^{k+1} \cdot n^{\mathcal{O}(1)}$.*

In the Disjoint Weighted-FVS, we are given a graph G, a weight function $w : V(G) \to \mathbb{Q}$, an integer k, and a feedback vertex set $R \subseteq V(G)$ of size $k + 1$. The objective is to find a set $X \subseteq V(G) \setminus R$ such that X is a minimum weighted-fvs of size at most k in G.

3.1. Reduction Rules for Disjoint Weighted-FVS

Let $(G, w : V(G) \to \mathbb{Q}, k, R)$ be an instance of Disjoint Weighted-FVS, and let $F = G - R$. We call a vertex $v \in V(F)$ a *nice* vertex if $d_G(v) = 2$ and both its neighbors are in R. A vertex $v \in V(F)$ is called *tent* if $d_G(v) = 3$ and all its neighbors are in R. We start with some simple reduction rules that preprocesses the graph. The Reduction Rules are applied in the order in which they are described.

Reduction Rule 3.1. *If $k < 0$ or $k = 0$ and G is not a forest, then return that (G, w, k, R) is a* No *instance of* Disjoint Weighted-FVS.

Reduction Rule 3.2. *If $k \geq 0$ and G is a forest, then return that (G, w, k, R) is a* Yes *instance of* Disjoint Weighted-FVS.

Reduction Rule 3.3. *If there is $v \in V(G)$ which is of degree one, then delete v. The resulting instance is $(G - \{v\}, w|_{V(G)\setminus\{v\}}, k, R \setminus \{v\})$.*

Reduction Rule 3.4. *If there is $v \in V(F)$ such that $G[R \cup \{v\}]$ has a cycle, then delete v from G and degrease k by 1. The resulting instance is $(G - \{v\}, w|_{V(G)\setminus\{v\}}, k - 1, R)$.*

Reduction Rule 3.5. *If there is an edge of multiplicity larger than 2 in $E(G)$, then reduce its multiplicity to 2.*

The correctness of the above Reduction Rules do not depend on the weights and thus it is similar to the one for undirected version of Weighted-FVS, and therefore, their safeness follows from [10].

Reduction Rule 3.6. *Let $x \in V(F)$ be a leaf with the only neighbor as y in F. Also, x has at most 2 neighbors in R. Subdivide the edge (x, y), and add the newly created vertex x^* to R. We define the new weight function $w^* : V(G) \cup \{x^*\} \to \mathbb{Q}$, as follows: $w^*(x^*) = 1$ and $w^*(v) = w(v)$, for $v \in V(G)$. Let G^* be the newly created graph after subdivision of the edge (x, y). Our reduced instance for* Disjoint Weighted-FVS *is $(G^*, w^*, k, R \cup \{x^*\})$.*

Lemma 4. *Reduction Rule 3.6 is safe.*

Proof. Let $x \in V(F)$ be a leaf with the only neighbor as y in F. Also, x has at most 2 neighbors in R. Let G^* be the graph after subdivision of the edge (x, y) with the newly added vertex as x^*. Furthermore, we define $w^* : V(G) \cup \{x^*\} \to \mathbb{Q}$, as follows: $w^*(x^*) = 1$ and $w^*(v) = w(v)$, for $v \in V(G)$. We will show that G has a weighted-fvs of size at most k of weight at most W if and only if G^* has a weighted-fvs of size at most k and weight at most W.

In the forward direction, let $S \subseteq V(G) \setminus R$ be a weighted-fvs of G of size at most k such that $w(S) \leq W$. We will show that indeed S is a weighted-fvs in G^* of size at most k and $w^*(S) \leq W$. We first bound the weight of S in G^*. Note that $w^*(v) = w(v)$, for $v \in V(G)$. Therefore $w^*(S) = w(S) \leq W$. We now show that S is a feedback vertex set in G^*. Suppose not, then there is a cycle C in $G^* - S$. If C does not contain x^*, then it contains none of the edges in $\{(x, x^*), (x^*, y)\}$. This by definition implies that C is also a cycle of $G - S$, which is a contradiction. On the other hand, let C contain the vertex x^*. By construction, C must also contain both (x, x^*) and (x^*, y). However, this means that $C' = (C - \{(x, x^*), (x^*, y)\}) \cup \{(x, y)\})$ is a cycle in $G - S$, which is a contradiction. Therefore, $S \subseteq V(G) \setminus R$ is also a weighted-fvs in G^*.

In the reverse direction, consider a weighted-fvs $S \subseteq V(G^*) \setminus (R \cup \{x^*\})$ of size at most k and $w^*(S) \leq W$. As S is a weighted-fvs disjoint from $R \cup \{x^*\}$, $x^* \notin S$. Thus, $w(S) = w^*(S)$, . We now show that S is a feedback vertex set in G. Suppose not, then there is a cycle C in $G - S$. If C does not contain the edge

(x, y), then by construction, C is also a cycle in $G^* - S$, which is a contradiction to the fact that S is a weighted-fvs of G^*. Otherwise, C contains the edge (x, y). But then, $C' = (C - \{(x, y)\}) \cup \{(x, x^*), (x^*, y)\}$ is a cycle $G^* - S$, which is a contradiction. Hence, S must is a weighted-fvs for G. $\square$

Now, we are ready to describe the main algorithm. To measure the running time of our algorithm for an instance $I = (G, w, k, R)$, we define the following measure.

$$\mu(I) = k + \rho(R) - (\eta + \tau)$$

Here, $\rho(R)$ is the number of connected components in $G[R]$ and η, τ are the number of nice vertices and tents in F, respectively.

Let $I = (G, w, k, R)$ be an instance of Disjoint Weighted-FVS where none of the Reduction Rules 3.1 to 3.6 are applicable. It is clear that Reduction Rules 3.1 to 3.5 do not increase the measure. We only need to argue about Reduction Rule 3.6, which increases the number of vertices in R. The number of vertices in R^* is one more than the number of vertices in R, and thus the number of connected components in $G^*[R^*]$ increases by (at most) one. However, we create either a *nice* vertex or a *tent* in G^*, therefore one of η or τ increases by 1. Hence, $\mu(I^*) = k + (\rho(R) + 1) - (\eta + \tau + 1) \leq \mu(I)$. Note that we do not increase the number of vertices in F, therefore we can apply the Reduction Rule 3.6 at most $|V(F)|$ times.

In Lemma 5, we show that if $\mu < 0$, then (G, w, k, R) is a No instance. This will form one of the base cases in our branching algorithm.

Lemma 5. *For an instance $I = (G, w, k, R)$ of* Disjoint Weighted-FVS*, if $\mu < 0$, then I is a* No *instance.*

Proof. Suppose that I is a Yes instance of Disjoint Weighted-FVS and $\mu < 0$. Let S be a weighted-fvs in G of size at most k, and $F' = G - S$, which is a forest. Let $N \subseteq V(G) \setminus R$, $T \subseteq V(G) \setminus R$ be the set of nice vertices and tents in $V(G) \setminus R$, respectively. Since F' is a forest we have that $G' = G[(R \cup N \cup T) \setminus S]$ is a forest. In G', we contract each of the connected components in R to a single vertex to obtain a forest $\tilde{F}$. Observe that $\tilde{F}$ has at most $|V(\tilde{F})| \leq \rho(R) + |N \setminus S| + |T \setminus S|$ vertices and thus can have at most $\rho(R) + |N \setminus S| + |T \setminus S| - 1$ many edges. The vertices in $(N \cup T) \setminus S \subseteq V(G) \setminus R$ forms an independent set in $\tilde{F}$, since they are nice vertices or tents. The vertices in $N \setminus S$ and $T \setminus S$ have degree 2 and degree 3 in $\tilde{F}$, respectively, since their degree cannot drop while contracting the components of $G[R]$. Thus we obtain the following.

$$2|N \setminus S| + 3|T \setminus S| \quad \leq |E(\tilde{F})| \leq \quad \rho(R) + |N \setminus S| + |T \setminus S| - 1$$

Therefore, $|N \setminus S| + |T \setminus S| < \rho(R)$. But $N \cap T = \emptyset$, and thus we have the following.

$$|N| + |T| < \rho(R) + |S| \leq \rho(R) + k \tag{1}$$

However, by our assumption, $\mu(I) = \rho(R) + k - (|N| + |T|) < 0$, and thus $|N| + |T| > \rho(R) + k$. This, contradicts the inequality given in Equation 1 contradicting our assumption that I is a YES instance of DISJOINT WEIGHTED-FVS. This completes the proof. $\square$

3.2. Algorithm for DISJOINT WEIGHTED-FVS

In this section, we conjure all that we have developed, and give the description of the algorithm for DISJOINT WEIGHTED-FVS, prove its correctness, and analyze its running time assuming a polynomial time procedure that we explain in the next subsection.

Description of the Algorithm. Let $I = (G, w, k, R)$ be an instance of DISJOINT WEIGHTED-FVS. If $G[R]$ is not a forest, then return that (G, w, k, R) is a NO instance of DISJOINT WEIGHTED-FVS. Hereafter, we will assume that $G[R]$ is a forest. First, the algorithm exhaustively applies the Reduction Rules 3.1 to 3.6. If at any point $\mu(I) < 0$, then we return that (G, w, k, R) is a NO instance. For sake of clarity, we will denote the reduced instance by (G, w, k, R). If all the vertices $v \in V(G) \setminus R$ are either nice vertices or tents then we solve the problem in polynomial time by using Theorem 9. We defer the proof of Theorem 9 to the following subsection, where we solve the instance using WEIGHTED MATROID PARITY. Otherwise, we apply the following Branching rule.

Branching Rule 3.1. If there is a leaf vertex $v \in V(G) \setminus R$, which is neither a nice vertex nor a tent then, we branch as follows.

i) v belongs to the solution. In this branch we delete v from G and decrease k by 1. The resulting instance is $(G - \{v\}, w', k-1, R)$, where $w' = w|_{V(G)\setminus\{v\}}$.
`The measure` μ `decreases by at least 1.`
ii) v does not belong to the solution. Note that v is neither a nice vertex nor a tent, and none of the reduction rules are applicable. Therefore, v has at least 3 neighbors in R. We add the vertex v to R. As a result, the number of components in $G[R]$ decreases by at least 2. The resulting instance is $(G, w, k, R \cup \{v\})$.
`The measure` μ `decreases by at least 2.`

The worst case branching vector corresponding to the above branching rule is $(1, 2)$.

Lemma 6. *The algorithm presented is correct.*

PROOF. Let $I = (G, w, k, R)$ be an instance of DISJOINT WEIGHTED-FVS. We prove the correctness of the algorithm by induction on the measure $\mu = \mu(I)$. By Lemma 5 when $\mu < 0$, then we correctly conclude that I is a NO instance.

For the induction hypothesis, let us assume that the algorithm correctly decides if the input is a YES/NO instance for $\mu = t$. We will prove it for $\mu = t + 1$. If any of the reduction rules are applicable, then either we correctly decide the instance or create an equivalent instance, which follows from their safeness. If we correctly decide the instance then the algorithm is trivially correct for $\mu = t+1$. Otherwise, we obtain an instance I'. If $\mu(I') < \mu(I)$ (the case when Reduction Rule 3.4 is applied) then by the induction hypothesis the algorithm correctly decides for the measure $\mu = t$. Otherwise, we have an instance with the same measure. If none of the reduction rules are applicable, then we have the following cases:

- Each $v \in V(G) \setminus R$ is either a nice vertex or a tent. In this case, we solve the problem in polynomial time, and the correctness of this step follows from Theorem 9.
- There is a leaf $v \in V(G) \setminus R$ such that v is neither a nice vertex nor a tent. The existence of such a vertex is guaranteed by the non-applicability of the reduction rules. In this case, we apply the Branching Rule 3.1. The Branching Rule is exhaustive. Moreover, at each branch the measure decreases at least by one. Hence, by the induction hypothesis it follows that the algorithm correctly decides whether or not I is a YES instance.

This completes the proof of correctness. $\square$

Next, we obtain an FPT algorithm for DISJOINT WEIGHTED-FVS using Lemma 6.

Lemma 7. DISJOINT WEIGHTED-FVS *can be solved in time* $\mathcal{O}^\star(2.618^k)$.

PROOF. The correctness of the algorithm follows from Lemma 6. All of the Reduction rules 3.1 to 3.6 can be applied in polynomial time. Also, at each branch we spend a polynomial amount of time. For each of the recursive calls at a branch, the measure μ decreases at least by 1. When $\mu < 0$, then we are able to correctly conclude that the given input is a no instance by Lemma 5. The number of leaves, and thus the size of the branching tree is upper bounded by the solution to the following recurrence.

$$T(\mu) \leq T(\mu - 1) + T(\mu - 2)$$

The above recurrence solves to 1.618^μ. Since at the start of the algorithm $\mu \leq 2k$, we have that the number of leaves is upper bounded by $\mathcal{O}(1.618^{2k})$. Therefore, DISJOINT WEIGHTED-FVS can be solved in time $\mathcal{O}^\star(2.618^k)$. $\square$

Using Lemma 7 and Observation 1, we obtain the prove Theorem 3.

3.3. Algorithm for sub-cubic DISJOINT WEIGHTED-FVS

Let (G, w, k, R) be an instance of DISJOINT WEIGHTED-FVS where each vertex in $V(G) \setminus R$ is either a nice vertex or a tent. An instance of the MATROID PARITY problem we create is same as that in [17]. In fact, what we use is the WEIGHTED MATROID PARITY problem.

The WEIGHTED MATROID PARITY problem for the graphic matroid $\mathcal{M}_H$ of a graph H is defined as follows. Let H be a graph with even number of edges, i.e. $|E(H)| = 2m$, where $m \in \mathbb{N}$, and we have a partition of $E(H)$ into pairs, say $E(H) = \{e_1^1, e_2^1\} \cup \{e_1^2, e_2^2\} \cup \cdots \cup \{e_1^m, e_2^m\}$. Furthermore, for each pair $\{e_1^i, e_2^i\}$, where $i \in [m]$ there is a positive (rational) weight $w_{\mathcal{M}}(\{e_1^i, e_2^i\})$. That is, $w_{\mathcal{M}}$ is a weight function on pairs. We want to find a set $I \subseteq [m]$ of maximum weight such that $\cup_{i\in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$. Equivalently, $\cup_{i\in I}\{e_1^i, e_2^i\}$ is acyclic in H. The Weighted Matroid Parity problem is polynomial time solvable on linear matroids, and hence in graphic matroids [20].

For each vertex $v \in V(G) \setminus R$, we arbitrarily label the edges incident to v. If v is a nice vertex then we label it as $\{e_1^v, e_2^v\}$; otherwise if v is a tent vertex then we label it as $\{e_0^v, e_1^v, e_2^v\}$. We let $w_{\mathcal{M}}(\{e_1^v, e_2^v\}) = w(v)$. Note that $F = E(G[R]) \cup \{e_0^v : v \in V(G) \setminus R\}$ is a forest. We contract all the edges in G which are in F to obtain a (new) graph H. In the process of contraction, we have not contracted any multiple edge or self loops. Also, we have $E(H) = \bigcup_{v\in V(G)\setminus R}\{e_1^v, e_2^v\}$. The input to the Weighted Matroid Parity algorithm for graphical matroid is the graph H, the set of pairs $\{e_1^v, e_2^v\}$ with weight $w_{\mathcal{M}}(\{e_1^v, e_2^v\})$, for $v \in V(G) \setminus R$. In Lemma 8 we prove that finding a minimum weighted-fvs $X \subseteq V(G) \setminus R$ in (G, w, k, R) is equivalent to computing a maximum weight subset $I \subseteq \{\{e_1^v, e_2^v\} \mid v \in V(G) \setminus R\}$ such that $\cup_{v\in I}\{e_1^v, e_2^v\}$ is an independent set in $\mathcal{M}_H$.

Lemma 8. *For a set $I \subseteq V(G) \setminus R$, $\cup_{i\in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$ of maximum weight if and only if $(V(G) \setminus R) \setminus I$ is a feedback vertex set in G of minimum weight.*

Proof. By the definition of H, $\cup_{i\in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$ if and only if $F \cup (\cup_{i\in I}\{e_1^i, e_2^i\})$ is acyclic in G. Recall that $F = E(G[R]) \cup \{e_0^v : v \in V(G) \setminus R\}$ is a forest. Therefore, if $\cup_{i\in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$, then $G' = G - ((V(G) \setminus R) \setminus I)$ is a forest. In other words, $(V(G) \setminus R) \setminus I$ is a feedback vertex set in G.

In the reverse direction, consider $I \subseteq V(G) \setminus R$ such that $(V(G) \setminus R) \setminus I$ is a feedback vertex set in G. This implies that $G[I \cup R]$ is a forest. Define $F' = \bigcup_{i\in I}\{e_1^i, e_2^i\}$. Clearly, $F' \subseteq E(G[I \cup R])$. Suppose, F' contains a cycle in H. This means that on uncontracting the edges in F, there is a cycle contained in $G[I \cup R]$, which is a contradiction. Therefore, $\cup_{i\in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$.

Note that by definition of $w_{\mathcal{M}}$, $w_{\mathcal{M}}(I) = w(I)$. Therefore, $w((V(G)\setminus R)\setminus I) = w(V(G)) - w(R) - w(I) = w(V(G)) - w(R) - w_{\mathcal{M}}(I)$. This implies that whenever $w_{\mathcal{M}}(I)$ is maximized then $w((V(G) \setminus R) \setminus I)$ is minimized and vice-versa. This completes the proof. $\square$

Lemma 8 immediately implies the following theorem.

Theorem 9. *Let (G, w, k, R) be an instance of* Disjoint Weighted-FVS. *If each $v \in V(G) \setminus R$ is either a nice vertex or a tent, then* Disjoint Weighted-FVS *in (G, w, k, R) can be solved in polynomial time.*

4. FPT algorithm for Block Graph Vertex Deletion

In this section, we design an FPT algorithm for the BGVD problem. First, we look at the special case, when the input graph does not have any small obstructions. Here, by small obstruction we mean either a $D_4 = K_4 - e$ or a C_4 (cycle on 4 vertices). We show that, in this case, BGVD reduces to Weighted-FVS. Later, we solve the general problem, using the algorithm of the special case.

4.1. Restricted BGVD

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

Restricted BGVD	**Parameter:** k
Input: A graph G, which is $\{D_4, C_4\}$-free and an integer k.	
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a block graph?	

Let (G, k) be an instance of Restricted BGVD, and $\mathcal{C}$ be the set of maximal cliques in G. We start with the following simple observation about graphs which is $\{D_4, C_4\}$-free.

Lemma 10. *For a $\{D_4, C_4\}$-free graph G on n vertices the following conditions hold.*

- *Any two maximal cliques intersect on at most one vertex.*
- *The number of maximal cliques in G is at most n^2.*

Proof. Let C_1 and C_2 be two distinct 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. □

Next, we construct an auxiliary weighted bipartite graph $\hat{G}$ as follows. The graph $\hat{G}$ 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 maximal clique $C \in \mathcal{C}$. Note that there is a bijective correspondence between the vertices of $V_{\mathcal{C}}$ and the maximal cliques in $\mathcal{C}$. A vertex $v \in C$, where $C \in \mathcal{C}$ is called a *shared vertex* if it is part of at least two (distinct) 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 a shared vertex of C.

Lemma 11. *Let G be a $\{D_4, C_4\}$-free graph, and $S \subseteq V(G)$. The set S is a block vertex deletion set of G if and only if $\hat{G} - S$ is acyclic.*

Proof. In the forward direction, let S be a block vertex deletion set for G. Suppose that $\hat{G} - S$ has a cycle C. From Lemma 10 if follows that C is not 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, $\hat{G} - S$ must be acyclic.

In the reverse direction, let $\hat{G} - S$ be acyclic. Suppose $G - S$ has an induced cycle C, of length at least four. As C is an induced cycle of length at least four, no two edges in 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 vertex corresponding to $C_{(u,v)}$ 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 an (induced) cycle C' in $\hat{G} - S$, which is a contradiction. Thus, S must be a block vertex deletion set for G. □

If the input graph G is without induced C_4 and D_4 then Lemma 11 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 $\hat{G}$ which is completely contained in $V(G)$ we solve an appropriate instance of Weighted-FVS. In particular, we set the weight function $w : V(\hat{G}) \to \mathbb{Q}$ 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 in $\hat{G}$ is at most n. This implies that using an algorithm for Weighted-FVS on the 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 12. Restricted BGVD *can be solved in* $\mathcal{O}^\star(3.618^k)$.

Proof. Let (G, k) be an instance of Restricted BGVD. We construct the instance $(\hat{G}, w, k)$ of Weighted-FVS as described earlier. Next, we solve Weighted-FVS on $(\hat{G}, w, k)$ using the algorithm given by Theorem 3 (Section 3). By Lemma 11 if $(\hat{G}, w, k)$ is a No instance of Weighted-FVS, then we correctly conclude that (G, k) is a No instance of Restricted BGVD. Otherwise, the algorithm return that $(\hat{G}, w, k)$ is a Yes instance of Restricted BGVD. Recall that by the construction of $(\hat{G}, w, k)$, for any solution S to the weighted-fvs in it we have $|S| \leq k$ and $w(S) \leq k$. This together with Lemma 11 implies that we correctly concluded that (G, k) is a Yes instance of Restricted BGVD. 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. □

4.2. Block Graph Vertex Deletion

We are now ready to describe the 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 (G, k) be an instance of BGVD, where G is a graph on n vertices. Furthermore, let O be an (induced) D_4 or C_4 (if any) in the input graph G. For any solution to BGVD in (G, k) must contain at least one of the vertices in O. 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 $(G - \{v\}, k - 1)$. Observe that (G, k) is a Yes instance of BGVD if and only if for some $v \in V(O)$ we have $(G - \{v\}, k - 1)$ is a Yes instance of BGVD. 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 given by Theorem 12 on (G, k), and return the same output. Thus, the running time of the algorithm is upper bounded by the following recurrence.

$$T(n,k) = \begin{cases} 3.168^k & \text{if G is } \{D_4, C_4\}\text{-free} \\ 4T(n-1,k-1) + n^{\mathcal{O}(1)} & \text{otherwise} \end{cases}$$

Thus, the running time of the algorithm is upper bounded by $\mathcal{O}^\star(4^k)$. $\square$

5. An Approximation Algorithm for BGVD

In this section, we present a (simple) approximation algorithm 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.

Theorem 13. BGVD *admits a factor four approximation algorithm.*

Proof. Let G be an 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. Furthermore, let Approx-WFVS be the 2-approximation algorithm given in [22], which takes as an instance $(G, w : V(G) \rightarrow \mathbb{Q})$, where G is a graph, and outputs a weighted-fvs whose weight is at most twice the weight of the minimum weighted-fvs.

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 $G' = G - S_1$. Observe that G' does not contain either D_4 or C_4 as an induced subgraph. Next, we construct a graph $\hat{G}'$ and a (vertex) weight function w, as described in Section 4.1, where we want to solve the problem of finding a weighted-fvs. We now employ the factor two approximation algorithm Approx-WFVS on the instance $(\hat{G}', w)$. This returns an weighted-fvs S_2 of $\hat{G}'$ such that $w(S_2)$ is at most twice the weight of a minimum weighted-fvs. By our construction $S_2 \subseteq V(G')$. Lemma 11 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 $S_{\mathsf{OPT}} - S_1$ is also an optimum solution for G' we have that $|S_2| \leq 2|S_{\mathsf{OPT}} - S_1|$.

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 the following.

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

This completes the proof. □

6. 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 BGVD, where G is a graph on n vertices. We start by applying some of the reduction rules from [13].

Reduction Rule 6.1. *If G has a component H, where H is a block graph, then remove H from G.*

Reduction Rule 6.2. *If there is a vertex $v \in V(G)$ such that $G - \{v\}$ has a component H, where $G[\{v\} \cup V(H)]$ is a connected block graph then, remove all the vertices in H from G.*

Reduction Rule 6.3. *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 6.4. *Let (t_1, t_2, t_3, t_4) be an induced path in G. For each $i \in [3]$, 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 $i \in [3]$, and $v \in S_i$ we have $N_G(v) \setminus S_i = \{t_i, t_{i+1}\}$, and*
- *For each $i \in \{2, 3\}$, we have $N_G(t_i) = \{t_{i-1}, t_{i+1}\} \cup S_{i-1} \cup S_i$.*

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

Proposition 1 (Proposition 3.1 [13]). *Let G be a graph and k be an 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, or*
iii) *$S'_v \subseteq V(G)$ such that $|S'_v| \leq 7k$ and $G - S'_v$ has no block graph obstruction containing v.*

Reduction Rule 6.5. *Let $v \in V(G)$ and $G' = G - \{v\}$. We remove the edges between $N_G(v)$ from G', i.e. $E(G') = E(G') \setminus \{(u, w) \mid u, w \in N_G(v)\}$. 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 (G,k) is a no-instance.*
- *Otherwise, delete v from G and decrease k by 1, i.e., the resulting instance is $(G-\{v\},k-1)$.*

The Reduction rules 6.1 to 6.5 are safe and can be applied in polynomial time [13]. 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. Hereafter, we assume that Reduction rules 6.1 to 6.5 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 6.5 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 $G - S'_v$ does not contain any obstruction that contains 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, which does not contain v. Let A be an approximate solution of size at most $4k$ (if it exists) to BGVD obtained by using the approximation algorithm for BGVD given by Theorem 13. If the approximation algorithm returns a solution whose size is more than $4k$, then we can trivially return that (G,k) is a No instance of BGVD. Therefore, in the following we assume that set A exists.

Observation 2. *For every vertex $v \in V(G)$, we can find in polynomial time, a set $S_v \subseteq V(G) \setminus \{v\}$ such that $|S_v| \leq 11k$ and $G - S_v$ is a block graph.*

Proof. If $v \notin A$, then $S_v = A$. Otherwise, $S_v = (A \setminus \{v\}) \cup S'_v$, where S'_v is the set given by Proposition 1. Here, we rely on the fact that Reduction Rule 6.5 is not applicable. Note that for each $v \in V(G)$, we have $|S_v| \leq 11k$ and $G - S_v$ is a block graph. $\square$

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 $G - (S_v \cup \{v\})$ 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* [10].

For $q \geq 1$, a *q-star* 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 to 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 14 (Expansion Lemma [10]). *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.

For a vertex $v \in V(G)$, let $\mathcal{C}_v$ be the set of connected components in $G-(S_v \cup \{v\})$ that have a vertex adjacent to v. Consider a connected component $C \in \mathcal{C}_v$ such that no vertex in $V(C)$ is adjacent to a vertex in S_v. But then, $G-\{v\}$ has a component which is a block graph (namely, C) therefore, Reduction rule 6.2 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 which contains a vertex d corresponding to each component $D \in \mathcal{C}$. Consider the bipartite graph $\mathcal{B}_v$ with the vertex set bipartition 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 $(u_d, s) \in E(G)$. Next, we state our final reduction rule.

Reduction Rule 6.6. *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 14 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 6.6 follows from the safeness of Reduction rule 6 in [13].

6.1. Bounding the number of blocks in $G - A$

First, we bound the number of leaf blocks in $G - A$, when none of the reduction rules are applicable. Note that $G - A$ is a block graph as 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 2. Let $\mathcal{C}_v$ be the set of connected components in $G - (S_v \cup \{v\})$ which have a vertex adjacent to v. All the connected components in $G - A$ which do not have a vertex that is adjacent to v, must be adjacent to some $v' \in A$ otherwise, Reduction rule 6.1 will be applicable. Also, all the leaf blocks in $G - A$ must have an internal vertex that is adjacent to some vertex in A, since the Reduction rules 6.1 and 6.2 are not applicable. The number of leaf blocks in $G - A$, 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 Observation 3, we give a bound on the number of leaf blocks in $G - A$, not containing any vertex from S'_v.

Observation 3. *For $v \in A$, the number of leaf blocks in $G - A$ not containing any vertex from S'_v is at most the number of leaf blocks in $G - (S_v \cup \{v\})$.*

PROOF. Note that for $v \in A$, $S_v = (A \setminus \{v\}) \cup S'_v$. By deleting a vertex $u \in S'_v$ from $G - A$ one of the following can happen. If u was a cut vertex in $G - A$, then we increase the number of components after deleting u from $G - A$. By increasing the number of components, the number of leaves cannot decrease. If u was not a cut vertex in $G - A$ then by deleting u the number of cut vertices can only increase. Therefore, the number of leaf blocks after deletion of u from $G - A$ can only increase. Hence, the claim follows. $\square$

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

PROOF. Consider a vertex $v \in V(G)$, and let $\mathcal{C}$ be the set of connected components in $G - (S_v \cup \{v\})$. By the definition of S_v, for each $C \in \mathcal{C}$, $G[V(C) \cup \{v\}]$ 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 a path with shortest path length between t and t'.

We first argue for the case when $(t, t') \notin E(G)$. In this case, 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 $G[\{t, t', u, v\}]$ is a D_4, otherwise it is a C_4, contradicting that $G[V(C) \cup \{v\}]$ is a block graph.

Let us assume that we can find an obstruction if the path length is ℓ. Next, we prove it for paths of length $\ell+1$. Let $P = (t, x_1, x_2, \ldots, x_{\ell-1}, t')$, 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 $G[V(C) \cup \{v\}]$ is a block graph. If $y = x_1$, then $G[\{t, x_1, x_2, v\}]$ either is a D_4, which is 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 thus by induction hypothesis has an obstruction along with v. Otherwise, $y \notin \{x_1, t'\}$, and the path $P' = (t, x_1, \ldots, y)$ is a path of length less than ℓ, which has at least 2 edges with $(y, v) \in E(G)$. Therefore, by induction hypothesis there is an obstruction along with the vertex v, contradicting that $G[V(C) \cup \{v\}]$ 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 (and $G - (S_v \cup \{v\})$ is a block graph). If there is an edge, say (t, u), then $G[\{t, t', u, v\}]$ is a D_4, contradicting that $G[V(C) \cup \{v\}]$ 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 $N_C(v) \subseteq V(\tilde{B})$. $\square$

Lemma 16. *For every $v \in A$, the number of leaf blocks in $\mathcal{C}_v$ is at most $\mathcal{O}(k)$.*

Proof. Every leaf block must have at least one internal vertex. By Lemma 15 we know that neighbours of v are contained in a block of C, where $C \in \mathcal{C}_v$. Therefore, v cannot be adjacent to internal vertices of two leaf blocks in $\mathcal{C}_v$. In other words, v can be adjacent to vertices in at most one leaf block from $\mathcal{C}_v$. But $|\mathcal{C}_v| \leq 33k$, since the Reduction rule 6.6 is not applicable. Therefore, the number of leaf blocks in $G - (S_v \cup \{v\})$ in which v has a neighbour is at most $\mathcal{O}(k)$. $\square$

Observe that in $G - A$, a vertex $v \in A$ can be adjacent to at most $\mathcal{O}(k)$ leaf blocks by Observation 3 and Lemma 16. Also, for a leaf block B in $G - A$, 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 6.2 is not applicable. Therefore, the number of leaf blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$.

Lemma 17. *The number of blocks B in $G - A$ such that the vertex set of B intersects with the vertex set of at least three other block in $G - A$ is $\mathcal{O}(k^2)$.*

Proof. Consider the block forest F_A of the block graph $G - A$. The number of blocks in $G - A$ is at most the number of vertices in F_A. Note that the set of leaves in F_A corresponds to the set of blocks in $G - A$ with at most one cut vertex. The number of leaf blocks is $G - A$ is bounded by $\mathcal{O}(k^2)$, and therefore the number of leaves in F_A is $\mathcal{O}(k^2)$. For a forest the number of vertices of degree at least 3 is bounded by the number of leaves (minus 2). Therefore, the number of degree three vertices in F_A is bounded by $\mathcal{O}(k^2)$. For a block B in $G - A$ which has at least 3 cut vertices, the vertex b corresponding to block B in F_A will be of degree at least 3. Therefore, the number of blocks in $G - A$ with at least three cut vertices is bounded by $\mathcal{O}(k^2)$.

Consider a block B in $G - A$ such that B has exactly 2 cut vertex, but $V(B)$ intersects with at least three blocks in $G - A$. Let b be the vertex corresponding to B in F_A, and v_B, u_B the cut-vertices in B. At least one of v_B, u_B is a cut-vertex in at least two blocks other than B, say v_B is such a cut vertex. If we contract the edge (b, v_B) in F_A to obtain F'_A, then number of leaves and degree 3 vertices in F'_A remains the same, which is bounded by $\mathcal{O}(k^2)$. Furthermore, the contracted vertex b^* is of degree at least 3. There is a bijection f between the vertices corresponding to blocks in F_A and F'_A, where $f(b) = b^*$ and $f(b') = b'$, for all $b' \in V_B(F_A) \setminus \{b\}$, where $V_B(F_A)$ is the set of vertices corresponding to blocks in $G - A$. Observe that this together with the previous argument implies that the number of blocks whose vertex set intersects with vertex set of at least 3 other blocks is bounded by $\mathcal{O}(k^2)$. $\square$

Let $\mathcal{L}$ be the set of leaf blocks in $G - A$ and $\mathcal{T}$ be the set of blocks in $G - A$ such that each block in $\mathcal{T}$ intersects with at least three other blocks in $G - A$. By Lemmas 16 and 17, we have that $|\mathcal{L}| \in \mathcal{O}(k^2)$ and $|\mathcal{T}| \in \mathcal{O}(k^2)$.

Let B be a block in $G^* = G - (S_v \cup \{v\})$ such that the vertex set of B has exactly two cut vertices, intersects with exactly two blocks of G^*, and it does not intersect with blocks in $\mathcal{L} \cup \mathcal{T}$. Also, B has a vertex that is a neighbor of v. We call such blocks as *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 18. *Let $G^* = G - (S_v \cup \{v\})$. Then G^* has at most $\mathcal{O}(k)$ nice degree two blocks of v.*

Proof. Recall that $\mathcal{C}_v$ is the set of connected components in $G - (S_v \cup \{v\})$ which have a vertex adjacent to v. From Lemma 15, for each of the connected component $C \in \mathcal{C}_v$, there is a block $\tilde{B}$ in C such that $N_C(v) \subseteq V(\tilde{B})$. Consider two nice degree two blocks B and B' in C of v. Let $b \in V(B)$ and $b' \in V(B')$ be the vertices such that $(v,b),(v,b') \in E(G)$. Next, we consider the following cases.

- Consider the case when $b \neq b'$. In this case, $b, b' \in V(\tilde{B})$, where $N_C(v) \subseteq V(\tilde{B})$ and $\tilde{B}$ is a block with exactly 2 cut vertices. Consider a block $\hat{B}$ in C other than B, B', and $\tilde{B}$. Notice that $V(\hat{B}) \cap V(\tilde{B}) = \emptyset$, and therefore, v cannot be adjacent to any vertex in $\hat{B}$. Hence, it follows that the number of blocks in C that contain a neighbor of v is $\mathcal{O}(1)$.

- Next, consider the case when $b = b'$. In this case, b is a cut vertex in both B and B'. Recall that B, B' both have exactly two cut vertices and intersect exactly two blocks in C. Therefore, one of B, B' must be same as $\tilde{B}$, say $B = \tilde{B}$. But B shares cut vertices with exactly two blocks. Therefore, v can have neighbors in at most $\mathcal{O}(1)$ blocks in C.

But recall that $|\mathcal{C}_v| = \mathcal{O}(k)$. Hence, the claim follows. $\square$

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

Lemma 19. *The number of blocks in $G - A$ with exactly two cut vertices is bounded by $\mathcal{O}(k^2)$.*

Proof. From Lemma 16, the number of leaf blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$. Let F_A be the block forest of the block graph $G - A$. From Lemmata 16 and 17, we know that $|\mathcal{L} \cup \mathcal{T}| \in \mathcal{O}(k^2)$. Also, from Lemma 18 the number of blocks in $\mathcal{T}_1$ is bounded by $\mathcal{O}(k^2)$.

Let $\mathcal{P}$ be the set of paths in F_A such that the endpoints are vertices corresponding to blocks in $\mathcal{L} \cup \mathcal{T} \cup \mathcal{T}_1$ and all internal block vertices do not correspond to blocks in $\mathcal{L} \cup \mathcal{T} \cup \mathcal{T}_1$. Note that, all internal vertices of such paths have degree exactly two in F_A. Since F_A is a tree, the number of paths in $\mathcal{P}$ is bounded by $\mathcal{O}(k^2)$.

Let $\mathcal{T}_2$ be the set of blocks in $G - A$ which have exactly two cut vertices and are not in $\mathcal{T}_1$. By definitions of F_A and $\mathcal{P}$, the vertex corresponding to a block $B \in \mathcal{T}_2$ must be an internal vertex in some path of $\mathcal{P}$. Consider a path P_A in $\mathcal{P}$. Note that F_A is a bipartite graph with the vertex bipartition as $(\mathcal{B}, V_C)$, where $\mathcal{B}$ is the set of blocks in $G - A$ and V_C is the set of cut vertices in $G - A$. Therefore, in P no two cut vertices in $V(G - A)$ can be adjacent to each other. Similarly, for $b, b' \in \mathcal{B}$, $(b, b') \notin E(P)$. Let $\mathcal{S}$ be the sequence of blocks in the order they appear in the path P_A. In $\mathcal{S}$, every consecutive blocks share a cut vertex. We remove the starting block and the end block from $\mathcal{S}$. We do this because these blocks belong to $\mathcal{L} \cup \mathcal{T} \cup \mathcal{T}_1$. If $\mathcal{S}$ has more than two blocks then, Reduction rule 6.4 would be applicable. Therefore, the number of blocks in $\mathcal{P}$ can be at most $\mathcal{O}(1)$.

The set of blocks in $G - A$ with exactly two cut vertices is contained in $\mathcal{T} \cup \mathcal{T}_1 \cup \mathcal{T}_2$. Hence, it follows that the number of blocks with exactly 2 cut vertices in $G - A$ is bounded by $\mathcal{O}(k^2)$. □

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

Lemma 20. *Consider a graph G, a positive integer k, and an approximate block vertex deletion set A of size $\mathcal{O}(k)$. If none of the Reduction rules 6.1 to 6.6 are applicable then the number of blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$.*

Proof. Follows from Lemmas 16, 17 and 19. □

6.2. 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 $G - A$. Consider a block B in $G - A$. We partition the set of internal vertices $V_I(B)$ of block B into three sets namely, $\mathcal{B}, \mathcal{R}$, and $\mathcal{I}$ depending on the neighborhood of A in B. We also partition the vertices in A depending on the number of vertices they are adjacent to in B. In Lemma 21 we show that the number of internal vertices in a block in $G - A$ 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 21. *Let (G, k) be an instance to* BGVD *and A be an approximate block vertex deletion set of G of size $\mathcal{O}(k)$. If none of the Reduction rules 6.1 to 6.6 are applicable then the number of internal vertices in a block B of $G - A$ is bounded by $\mathcal{O}(k^2)$.*

Proof. Let $A_{\leq 2k+1} = \{v \in A \mid |N_B(v)| \leq 2k + 1\}$ and $A_{>2k+1} = A \setminus A_{\leq 2k+1}$. For a vertex $u \in V_I(B)$, if u is adjacent to at least one of the vertices in $A_{\leq 2k+1}$

then, we add u to the set $\mathcal{B}$. Note that the number of vertices in $\mathcal{B}$ is bounded by $\mathcal{O}(k^2)$, since each $v \in A_{\leq 2k+1}$ is adjacent to at most $2k+1$ vertices in $V_I(B)$. Also, for a vertex $u \in V_I(B) \setminus \mathcal{B}$, $N(u) \cap A_{\leq 2k+1} = \emptyset$.

For each vertex $u \in V_I(B) \setminus \mathcal{B}$, if u is not adjacent to at least one vertex in $A_{>2k+1}$ then, we add u to the set $\mathcal{R}$. For $v \in A_{>2k+1}$, let Q_v be the set of vertices in $V_I(B) \setminus \mathcal{B}$ which are not adjacent to v. Note that $|Q_v| \leq k$ otherwise, for each pair of vertices $t_1, t_2 \in N_B(v)$ along with one vertex in Q_v we get $k+1$ vertex disjoint obstruction, namely D_4, intersecting only at v, which contradicts the non-applicability of Reduction rule 6.5. Therefore, the number of vertices in $\mathcal{R}$ is bounded by $\mathcal{O}(k^2)$.

Let $\mathcal{I} = V_I(B) \setminus (\mathcal{B} \cup \mathcal{R})$. Note that the vertices in $\mathcal{I}$ induce a clique. Furthermore, for each $w \in \mathcal{I}$, $N(w) \cap A_{\leq 2k+1} = \emptyset$ and $N(w) \cap A_{>2k+1} = A_{>2k+1}$. Therefore, each $w, w' \in \mathcal{I}$ are twins. In fact, $\mathcal{I}$ is a set of twins. If $|\mathcal{I}| > k+1$ then Reduction rule 6.3 would be applicable. Therefore, $|\mathcal{I}| \leq k+1$. But, $|V_I(B)| = |\mathcal{B}| + |\mathcal{R}| + |\mathcal{I}|$, which 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 2.

Proof (of Theorem 2). 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 6.1 to 6.6 are applicable. By Lemma 20, the number of blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$. From Lemma 21, the number of internal vertices in a block of $G - A$ is bounded by $\mathcal{O}(k^2)$. Also note that the number of cut-vertices in $G - A$ is bounded by the number of blocks in $G - A$, i.e. $\mathcal{O}(k^2)$. The number of vertices in $G - A$ is sum of the number of internal vertices in $G - A$ and the number of cut vertices in $G - A$. Therefore, $|V(G)| = |V(G - A)| + |A| = (\mathcal{O}(k^2) \cdot \mathcal{O}(k^2) + \mathcal{O}(k^2)) + \mathcal{O}(k) = \mathcal{O}(k^4)$. □

References

[1] J. M. Lewis, M. Yannakakis, The node-deletion problem for hereditary properties is np-complete, Journal of Computer and System Sciences 20 (1980) 219 – 230.

[2] F. V. Fomin, D. Lokshtanov, N. Misra, S. Saurabh, Planar F-Deletion: Approximation, Kernelization and Optimal FPT algorithms, in: 53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS), 2012, pp. 470–479.

[3] T. Fujito, A unified approximation algorithm for node-deletion problems, Discrete Applied Mathematics 86 (1998) 213–231.

[4] C. Lund, M. Yannakakis, On the hardness of approximating minimization problems, Journal of the ACM 41 (1994) 960–981.

[5] D. Marx, B. O'Sullivan, I. Razgon, Finding small separators in linear time via treewidth reduction, ACM Transactions on Algorithms 9 (2013) 30:1–30:35.

[6] R. G. Downey, M. R. Fellows, Parameterized complexity, Springer-Verlag, 1997.

[7] R. G. Downey, M. R. Fellows, Fundamentals of Parameterized complexity, Springer-Verlag, 2013.

[8] J. Flum, M. Grohe, Parameterized Complexity Theory, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2006.

[9] R. Niedermeier, Invitation to fixed-parameter algorithms, Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press, 2006.

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

[11] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, S. Sikdar, Linear kernels and single-exponential algorithms via protrusion decompositions, ACM Transactions on Algorithms 12 (2015) 21:1–21:41.

[12] M. M. Kanté, E. J. Kim, O. Kwon, C. Paul, An FPT algorithm and a polynomial kernel for linear rankwidth-1 vertex deletion, Algorithmica 79 (2017) 66–95.

[13] E. J. Kim, O. Kwon, A polynomial kernel for block graph deletion, Algorithmica 79 (2017) 251–270.

[14] A. Brandstädt, V. B. Le, J. P. Spinrad, Graph Classes: A Survey, Society for Industrial and Applied Mathematics, 1999.

[15] E. Howorka, A characterization of ptolemaic graphs, Journal of Graph Theory 5 (1981) 323–331.

[16] E. J. Kim, O. Kwon, A polynomial kernel for block graph deletion, in: 10th International Symposium on Parameterized and Exact Computation, (IPEC), 2015, pp. 270–281.

[17] T. Kociumaka, M. Pilipczuk, Faster deterministic feedback vertex set, Information Processing Letters 114 (2014) 556–560.

[18] S. Thomassé, A $4k^2$ kernel for feedback vertex set, ACM Transactions on Algorithms 6 (2010) 32:1–32:8.

[19] J. Chen, F. V. Fomin, Y. Liu, S. Lu, Y. Villanger, Improved algorithms for feedback vertex set problems, Journal of Computer and System Sciences 74 (2008) 1188 – 1198.

[20] S. Iwata, Y. Kobayashi, A weighted linear matroid parity algorithm, in: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), 2017, pp. 264–276.

[21] R. Diestel, Graph Theory, 4th Edition, volume 173 of *Graduate texts in mathematics*, Springer, 2012.

[22] V. Bafna, P. Berman, T. Fujito, A 2-approximation algorithm for the undirected feedback vertex set problem, SIAM Journal on Discrete Mathematics 12 (1999) 289–297.