Conflict Free Feedback Vertex Set: A Parameterized Dichotomy

Akanksha Agrawal
Institute of Mathematical Sciences, HBNI, Chennai, India
akanksha.agrawal.2029@gmail.com

Pallavi Jain
Institute of Mathematical Sciences, HBNI, Chennai, India
pallavij@imsc.res.in

Lawqueen Kanesh
Institute of Mathematical Sciences, HBNI, Chennai, India
lawqueen@imsc.res.in

Daniel Lokshtanov
University of Bergen, Bergen, Norway
daniello@ii.uib.no

Saket Saurabh
Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Abstract

In this paper we study recently introduced conflict version of the classical Feedback Vertex Set (FVS) problem. Let $\mathcal{F}$ be a family of graphs. Then, for every family $\mathcal{F}$, we get $\mathcal{F}$-CF-Feedback Vertex Set ($\mathcal{F}$-CF-FVS, for short). The problem $\mathcal{F}$-CF-FVS takes as an input a graph G, a graph $H \in \mathcal{F}$, and an integer k, and the objective is to decide if there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest and S is an independent set in H. Observe that if we instantiate $\mathcal{F}$ to be the family of edgeless graphs then we get the classical FVS problem. Jain, Kanish and Misra [CSR 2018] showed that in contrast to FVS, $\mathcal{F}$-CF-FVS is W[1]-hard on general graphs and admits an FPT algorithm if $\mathcal{F}$ is a family of d-degenerate graphs. In this paper we relate $\mathcal{F}$-CF-FVS to the Independent Set problem on special classes of graphs and obtain a complete dichotomy result on the Parameterized Complexity of the problem $\mathcal{F}$-CF-FVS. In particular, we show that $\mathcal{F}$-CF-FVS is FPT parameterized by the solution size if and only if $\mathcal{F}$+Cluster IS is FPT parameterized by the solution size. Here, $\mathcal{F}$+Cluster IS is the Independent Set problem in the (edge) union of a graph $G \in \mathcal{F}$ and a cluster graph H (G and H are explicitly given). Next we exploit this characterization to obtain new FPT results as well as intractability results for $\mathcal{F}$-CF-FVS. In particular, we give FPT algorithms for $\mathcal{F}$+Cluster IS when $\mathcal{F}$ is the family of $K_{i,j}$-free graphs. Finally, we consider for each $0 < \epsilon < 1$, the family of graphs $\mathcal{F}_\epsilon$, which comprise of graphs G such that $|E(G)| \leq |V(G)|^{2-\epsilon}$ and show that $\mathcal{F}_\epsilon$+Cluster IS is W[1]-hard, when parameterized by the solution size, for every $0 < \epsilon < 1$.

2012 ACM Subject Classification Dummy classification

Keywords and phrases Dummy keyword

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.23

1 Introduction

Feedback Vertex Set (FVS) is one of the classical NP-hard problems that has been subjected to intensive study in algorithmic paradigms that are meant for coping with NP-hard problems, and particularly in the realm of Parameterized Complexity. In this problem, given a graph G and an integer k, the objective is to decide if there is $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest. FVS has received a lot of attention in the realm of Parameterized Complexity. This problem is known to be in FPT, and the best known algorithm for it runs in time $\mathcal{O}(3.618^k n^{\mathcal{O}(1)})$ [7, 14]. Several variant and generalizations of Feedback Vertex Set such as Weighted Feedback Vertex Set [2, 6], Independent Feedback Vertex Set [1, 16], Connected Feedback Vertex Set [17], and Simultaneous Feedback Vertex Set [3, 5] have been studied from the viewpoint of Parameterized Complexity.

Recently, Jain et al. [13] defined an interesting generalization of well-studied vertex deletion problems – in particular for FVS. The CF-Feedback Vertex Set (CF-FVS, for short) problem takes as input graphs G and H, and an integer k, and the objective is to decide if there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest and S is an independent set in H. The graph H is also called a *conflict graph*. Observe that the CF-FVS problem generalizes many classical graph problems such as Feedback Vertex Set, Independent Feedback Vertex Set, etc. Among other results, Jain et al. [13] showed that CF-FVS is W[1]-hard on general graphs. Also, they designed FPT algorithms when the input graph H is from the family of d-degenerate graphs and the family of nowhere dense graphs.

A natural way of defining CF-FVS will be by fixing a family $\mathcal{F}$ from which the conflict graph H is allowed to belong. Thus, for every fixed $\mathcal{F}$ we get a new CF-FVS problem. In particular we get the following problem.

$\mathcal{F}$-CF-Feedback Vertex Set ($\mathcal{F}$-CF-FVS) **Parameter:** k
Input: A graph G, a graph $H \in \mathcal{F}$ (where $V(G) = V(H)$), and an integer k.
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest and S is an independent set in H?

Jain et al. [13] showed that $\mathcal{F}$-CF-FVS is W[1]-hard when $\mathcal{F}$ is family of all graphs and admits FPT algorithm when the input graph H is from the family of d-degenerate graphs and the family of nowhere dense graphs. The most natural question that arises here is the following.

Question 1: ***For which graph families $\mathcal{F}$, $\mathcal{F}$-CF-FVS is FPT?***

Our Results. Starting point of our research is Question 1. We obtain a complete dichotomy result on the Parameterized Complexity of the problem $\mathcal{F}$-CF-FVS in terms of another well-studied problem, namely, the Independent Set problem – the wall of intractability. Towards stating our results, we start by defining the problem $\mathcal{F}$+Cluster IS, which is of independent interest. A *cluster graph* is a graph formed from the disjoint union of complete graphs (or clique).

$\mathcal{F}$+Cluster Independent Set ($\mathcal{F}$+Cluster IS) **Parameter:** k
Input: A graph $G \in \mathcal{F}$, a cluster graph H (where $V(G) = V(H)$), and an integer k, such that H has exactly k connected components.
Question: Is there a set $S \subseteq V(G)$ of size k such that S is an independent set in both G and in H?

We note that $\mathcal{F}$+Cluster IS is the Independent Set problem on the edge union of two graphs, where one of the graphs is from a family of graphs $\mathcal{F}$ and the other one is a cluster graph. Here, additionally we know the partition of edges into two sets E_1 and E_2 such that the graph induced on E_1 is in $\mathcal{F}$ and the graph induced on E_2 is a cluster graph. We note that $\mathcal{F}$+CLUSTER IS has been studied in the literature for $\mathcal{F}$ being the family of interval graphs (with no restriction on the number of clusters) [21]. They showed the problem to be FPT. Recently, Bentert et al. [?] generalized the result from interval graphs to chordal graphs. This problem arises naturally in the study of scheduling problems. We refer the readers to [21, ?] for more details on the application of $\mathcal{F}$+CLUSTER IS.

We are now ready to state our results. We show that $\mathcal{F}$-CF-FVS is in FPT if and only if $\mathcal{F}$+CLUSTER IS is in FPT. This gives a complete characterization of when the $\mathcal{F}$-CF-FVS problem is in FPT. To prove the forward direction, i.e., showing that $\mathcal{F}$+CLUSTER IS is in FPT implies $\mathcal{F}$-CF-FVS is in FPT, we design a branching based algorithm, which at the base case generates instances of $\mathcal{F}$+CLUSTER IS, which is solved using the assumed FPT algorithm for $\mathcal{F}$+CLUSTER IS. Thus, we give "fpt-turing-reduction" from $\mathcal{F}$-CF-FVS to $\mathcal{F}$+CLUSTER IS. It is worth to note that there are very known reductions of this nature. To show that $\mathcal{F}$-CF-FVS is in FPT implies that $\mathcal{F}$+CLUSTER IS is in FPT, we give an appropriate reduction from $\mathcal{F}$+CLUSTER IS to $\mathcal{F}$-CF-FVS, which proves the statement.

Next, we consider two families of graphs. We first design FPT algorithm for the corresponding $\mathcal{F}$+CLUSTER IS problem. For the second class we give a hardness result. First, we consider the problem $K_{i,j}$-free+CLUSTER IS, which is the $\mathcal{F}$+CLUSTER IS problem for the family of $K_{i,j}$-free graphs. We design an FPT algorithm for $K_{i,j}$-free+CLUSTER IS based on branching together with solving the base cases using a greedy approach. This adds another family of graphs, apart from interval and chordal graphs, such that $\mathcal{F}$+CLUSTER IS is FPT.

We note that $K_{i,j}$-free graphs have at most $n^{2-\epsilon}$ edges, where n is the number of vertices in the input graph and $\epsilon = \epsilon(i,j) > 0$ [20, 12]. We complement our FPT result on $K_{i,j}$-free+CLUSTER IS with the W[1]-hardness result of the $\mathcal{F}$+CLUSTER IS problem when $\mathcal{F}$ is the family of graphs with at most $n^{2-\epsilon}$ edges. This result is obtained by giving an appropriate reduction from the problem MULTICOLORED BICLIQUE, which is known to be W[1]-hard [7, 10]. We also show that the $\mathcal{F}$+CLUSTER IS problem is W[1]-hard when $\mathcal{F}$ is the family of bipartite graphs. Again, this result is obtained via a reduction from MULTICOLORED BICLIQUE.

change introduction

add/ check citation for subquadratic edges in kij free graphs

setting epsilon in the last paragraph, check at the end.

2 Preliminaries

In this section, we state some basic definitions and terminologies from Graph Theory that are used in this paper. For the graph related terminologies which are not explicitly defined here, we refer the reader to the book of Diestel [8].

Graphs.

Consider a graph G. By $V(G)$ and $E(G)$ we denote the set of vertices and edges in G, respectively. When the graph is clear from the context, we use n and m to denote the number of vertices and edges in the graph, respectively. For $X \subseteq V(G)$, by $G[X]$ we denote the subgraph of G with vertex set X and edge set $\{uv \in E(G) \mid u, v \in X\}$. Moreover, by $G - X$ we denote graph $G[V(G) \setminus X]$. For $v \in V(G)$, $N_G(v)$ denotes the set $\{u \mid uv \in E(G)\}$, and $N_G[v]$ denotes the set $N_G(v) \cup \{v\}$. By $\mathsf{deg}_G(v)$ we denote the size of $N_G(v)$. A *path* $P = (v_1, \ldots, v_n)$ is an ordered collection of vertices, with endpoints v_1 and v_n, such that there is an edge between every pair of consecutive vertices in P. A *cycle* $C = (v_1, \ldots, v_n)$ is a path with the edge v_1v_n. Consider graphs G and H. We say that G is an H-*free* graph if no subgraph of G is isomorphic to H. For $u, v \in V(G) \cap V(H)$, we say that u and v are in *conflict* in G with respect to H if $uv \in E(H)$.

A *clique* is a subgraph of an undirected graph such that every two distinct vertices in it are adjacent. A *connected component* of an undirected graph is a (vertex) maximal induced subgraph in which every two vertices are connected by a path. If a graph has only one connected component then it is called *connected graph*. A graph is a *cluster* graph if each of its connected components are cliques. For $k \in \mathbb{N}$, a k-*cluster* graph is cluster graph with exactly k connected components. Let $\mathcal{C}$ be the set of connected components in cluster graph. We define vertex set of $\mathcal{C}$ as follows: $V(\mathcal{C}) = \cup_{C \in \mathcal{C}'} V(C)$. A graph G is a *complete bipartite* graph if its vertex set can be partitioned into two disjoint (independent) sets X and Y, such that $E(G) = \{xy \mid x \in X, y \in Y\}$. For $x, y \in \mathbb{N}$, by K_{xy} we denote the complete bipartite graph on $x + y$ vertices which admits a vertex bipartition into sets X and Y of sizes x and y, respectively, such that $E(K_{xy}) = \{xy \mid x \in X, y \in Y\}$. A graph is a *chordal* graph if it has no induced cycle of length at least 4.

Sets.

We denote the set of natural numbers and real numbers by $\mathbb{N}$ and $\mathbb{R}$, respectively. For $k \in \mathbb{N}$, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. For $a, b \in \mathbb{R}$, a *half open interval* denoted by $(a, b]$ is the set of all real numbers x, such that $a < x \leq b$. For a set X, by 2^X we denote the power set of X, i.e., the set comprising of all subsets of X.

Parameterized Complexity.

A *parameterized problem* Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ 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, called the *parameter*. A central notion in parameterized complexity is *fixed-parameter tractability* (or in FPT) which means, for a given instance (x, k), decidability in time $f(k) \cdot \mathsf{poly}(|x|)$, where $f(\cdot)$ is an arbitrary computable function and $\mathsf{poly}(\cdot)$ is a polynomial function. To prove that a problem is FPT, it is possible to give an explicit algorithm, called a *parameterized algorithm*, which solves it in time $f(k) \cdot \mathsf{poly}(|x|)$. On the other hand, to show that a problem is unlikely to be in FPT, it is possible to use FPT time reductions analogous to the polynomial time reductions employed in Classical Complexity. Here, the concept of W[t]-hardness replaces the concept of NP-hardness, and we need not only construct an equivalent instance in FPT time, but also ensure that the size of the parameter in the new instance depends only on the size of the parameter in the original instance. For more details on Parameterized Complexity, we refer the reader to the books of Downey and Fellows [9], Flum and Grohe [11], Niedermeier [18], and the recent book by Cygan et al. [7].

3 W-hardness of $\mathcal{F}$-CF-FVS Problems

This section is devoted to showing W-hardness results for $\mathcal{F}$-CF-FVS problems for certain graph classes, $\mathcal{F}$. In Section 3.1, we show one direction of our dichotomy result. That is, if for a family of graphs $\mathcal{F}$, $\mathcal{F}$+Cluster IS is not in FPT when parameterized by the size of solution then $\mathcal{F}$-CF-FVS is also not in FPT when parameterized by the size of solution. This result is obtained by giving a parameterized reduction from $\mathcal{F}$+Cluster IS to $\mathcal{F}$-CF-FVS. Next, we show that the problem $\mathcal{F}$-CF-FVS is W[1]-hard, when parameterized by the size of solution, where $\mathcal{F}$ is the family of bipartite graphs (Section 3.2) or the family of graphs with sub-quadratic number of edges (Section 3.3). These results are obtained by giving an appropriate reduction from the problem Multicolored Biclique, which is known to be W[1]-hard [7, 10].

3.1 $\mathcal{F}$+Cluster IS to $\mathcal{F}$-CF-FVS

In this section, we show that, for a family of graphs $\mathcal{F}$, if $\mathcal{F}$+Cluster IS is not in FPT, then $\mathcal{F}$-CF-FVS is also not in FPT (where the parameters are the solution sizes). To prove this result, we give a parameterized reduction from $\mathcal{F}$+Cluster IS to $\mathcal{F}$-CF-FVS.

Let (G, H, k) be an instance of $\mathcal{F}$+Cluster IS. We construct an instance (G', H', k') of $\mathcal{F}$-CF-FVS as follows. We have $H' = G$, $k' = k$, and $V(G') = V(H)$. Let $\mathcal{C}$ be the set of connected components in H. Recall that we have $|\mathcal{C}| = k$. For each $C \in \mathcal{C}$, we add a cycle (in an arbitrarily chosen order) induced on vertices in $V(C)$ in G'. This completes the description of the reduction. Next, we show the equivalence between the instance (G, H, k) of $\mathcal{F}$+Cluster IS and the instance (G', H', k') of $\mathcal{F}$-CF-FVS.

▶ **Lemma 1.** *(G, H, k) is a* yes *instance of $\mathcal{F}$+*Cluster IS *if and only if (G', H', k') is a* yes *instance of $\mathcal{F}$-*CF-FVS.

Proof. In the forward direction, let (G, H, k) be a yes instance of $\mathcal{F}$+Cluster IS, and S be one of its solution. Since $H' = G$, therefore, S is an independent set in H'. Let $\mathcal{C}$ be the set of connected components in H. As S is a solution, it must contain exactly one vertex from each $C \in \mathcal{C}$. Moreover, G' comprises of vertex disjoint cycles for each $C \in \mathcal{C}$. Thus S intersects every cycle in G'. Therefore, S is a solution to $\mathcal{F}$-CF-FVS in (G', H', k').

In the reverse direction, let (G', H', k') be a yes instance of $\mathcal{F}$-CF-FVS, and S be one of its solution. Recall that G' comprises of k vertex disjoint cycles, each corresponding to a connected component $C \in \mathcal{C}$, where $\mathcal{C}$ is the set of connected components in H. Therefore, S contains exactly one vertex from each $C \in \mathcal{C}$. Also, $H' = G$, and therefore, S is an independent set in G. This implies that S is a solution to $\mathcal{F}$+Cluster IS in (G, H, k). ◀

▶ **Theorem 2.** *For a family of graphs $\mathcal{F}$, if $\mathcal{F}$+*Cluster IS *is not in* FPT *when parameterized by the solution size, then $\mathcal{F}$-*CF-FVS *is also not in* FPT *when parameterized by the solution size.*

Proof. Follows from the construction of instance (G', H', k') of $\mathcal{F}$-CF-FVS for the given instance (G, H, k) of $\mathcal{F}$+Cluster IS with $H' = G$ and Lemma 1. ◀

3.2 W[1]-hardness on Bipartite Graphs

In this section, we show that for the family of bipartite graphs, $\mathcal{B}$, the $\mathcal{B}$-CF-FVS problem is W[1]-hard, when parameterized by the solution size. Throughout this section, $\mathcal{B}$ will denote the family of bipartite graphs. To prove our result, we give a parameterized reduction from the problem Multicolored Biclique to $\mathcal{B}$-CF-FVS. In the following, we formally define the problem Multicolored Biclique.

> Multicolored Biclique (MBC) **Parameter:** k
> **Input:** A bipartite graph G, a partition of A into k sets $A_1, A_2, \cdots, A_k$, and a partition of B into k sets $B_1, B_2, \cdots, B_k$, where A and B are a vertex bipartition of G.
> **Question:** Is there a set $S \subseteq V(G)$ such that for each $i \in [k]$ we have $|S \cap A_i| = 1$ and $|S \cap B_i| = 1$, and $G[S]$ is isomorphic to $K_{k,k}$?

Let $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ be an instance of Multicolored Biclique. We construct an instance (G', H', k') of $\mathcal{B}$-CF-FVS as follows. We have $V(G') = V(H') = V(G)$, and $E(H') = \{uv \mid u \in \cup_{i \in [k]} A_i, v \in \cup_{i \in [k]} B_i, \text{ and } uv \notin E(G)\}$. Next, for each $i \in [k]$, we add a cycle (in an arbitrary order) induced on vertices in A_i in G'. Similarly, we add for each $i \in [k]$, a cycle induced on vertices in B_i in G'. Notice that G' is comprises of $2k$ vertex disjoint cycles, and H' is a bipartite graph. Finally, we set $k' = 2k$. This completes the description of the reduction.

▶ **Lemma 3.** *$(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ is a* yes *instance of* Multicolored Biclique *if and only if (G', H', k') is a* yes *instance of* $\mathcal{B}$-CF-FVS.

Proof. In the forward direction, let $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ be a yes instance of Multicolored Biclique, and S be one of its solution. We will show that S is a solution to $\mathcal{B}$-CF-FVS in (G', H', k'). Since S is a solution to Multicolored Biclique in $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$, therefore for each $i \in [k]$, $|S \cap A_i| = 1$ and $|S \cap B_i| = 1$. Since G' comprises of vertex disjoint cycles corresponding to sets in A_i and B_i, therefore, S intersects every cycle in G'. By the construction of H' it follows that S is an independent set in H'. This concludes the proof of forward direction.

In the reverse direction, let (G', H', k') be a yes instance of $\mathcal{B}$-CF-FVS, and S be one of its solution. By construction of G', for each $i \in [k]$ we have $|S \cap A_i| = 1$ and $|S \cap B_i| = 1$ and by the construction of H', we have that S is isomorphic to $K_{k,k}$ in G. Therefore, S is a solution to Multicolored Biclique in $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$. ◀

Now we are ready to prove the main theorem of this section.

▶ **Theorem 4.** $\mathcal{B}$-CF-FVS *parameterized by the solution size is* W[1]*-hard, where $\mathcal{B}$ is the family of bipartite graphs.*

Proof. Follows from the construction, Lemma 3, and W[1]-hardness of Multicolored Biclique [7, 10]. ◀

3.3 W[1]-hardness on Graphs with Sub-quadratic Edges

In this section, we show that $\mathcal{F}$-CF-FVS is W[1]-hard, when parameterized by the solution size, where $\mathcal{F}$ is the family of graphs with sub-quadratic edges. To formalize the family of graphs with subquadratic edges, we define the following. For $0 < \epsilon < 1$, we define $\mathcal{F}_\epsilon$ to be the family comprising of graphs G, such that $|E(G)| \leq |V(G)|^{2-\epsilon}$. We show that for every $0 < \epsilon < 1$, the $\mathcal{F}_\epsilon$-CF-FVS problem is W[1]-hard, when parameterized by the solution size. Towards this, for each (fixed) $0 < \epsilon < 1$, we give a parameterized reduction from Multicolored Biclique to $\mathcal{F}_\epsilon$-CF-FVS.

Let $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ be an instance of Multicolored Biclique. We construct an instance (G', H', k') of $\mathcal{F}_\epsilon$-CF-FVS as follows. Let $n = |V(G)|$, $m = |E(G)|$, and X be a set comprising of $n^{\frac{2}{2-\epsilon}} - n$ (new) vertices. The vertex set of G' and H' is $X \cup V(G)$. For each $i \in [k]$, we add a cycle (in arbitrary order) induced on vertices in A_i in G'. Similarly, we add for each $i \in [k]$, a cycle induced on vertices in B_i in G'. Also, we add a cycle induced on vertices in X to G'. We have $E(H') = \{uv \mid u \in \cup_{i\in[k]} A_i, v \in \cup_{i\in[k]} B_i, \text{ and } uv \notin E(G)\}$. Finally, we set $k' = 2k + 1$. Notice that since $|V(H')| = n^{\frac{2}{2-\epsilon}}$, and $|E(H')| < n^2$, therefore, $H \in \mathcal{F}_\epsilon$.

▶ **Lemma 5.** *$(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ is a* yes *instance of* Multicolored Biclique *if and only if (G', H', k') is a* yes *instance of* $\mathcal{F}_\epsilon$-CF-FVS.

Proof. In the forward direction, let $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ be a yes instance of Multicolored Biclique, and S be one of its solution. Let $x \in X$ be an arbitrarily chosen vertex from X. We will show that $S \cup \{x\}$ is a solution to $\mathcal{F}_\epsilon$-CF-FVS in (G', H', k'). Since S is a solution to Multicolored Biclique in $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$, therefore for each $i \in [k]$, $|S \cap A_i| = 1$ and $|S \cap B_i| = 1$. Since G' comprises of vertex disjoint cycles corresponding to sets in A_i and B_i, and a cycle induced on vertices in X therefore, $S \cup \{x\}$ intersects every cycle in G'. By the construction of H' it follows that $S \cup \{x\}$ is an independent set in H'. This concludes the proof of forward direction.

In the reverse direction, let (G', H', k') be a yes instance of $\mathcal{F}_\epsilon$-CF-FVS, and S be one of its solution. Let $S' = S \setminus X$. By construction of G', for each $i \in [k]$ we have $|S' \cap A_i| = 1$ and $|S' \cap B_i| = 1$, and by construction of H', we have that S' is isomorphic to $K_{k,k}$ in G. Therefore, S' is a solution to Multicolored Biclique in $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$. ◀

Now we are ready to prove the main theorem of this section.

▶ **Theorem 6.** *For $0 < \epsilon < 1$, $\mathcal{F}_\epsilon$-CF-FVS parameterized by the solution size is* W[1]-hard.

Proof. Follows from the construction, Lemma 5, and W[1]-hardness of Multicolored Biclique [7, 10]. ◀

4 FPT algorithms for $\mathcal{F}$-CF-FVS for Restricted Conflict Graphs

For a hereditary (closed under taking induced subgraphs) family of graphs $\mathcal{F}$, we show that if $\mathcal{F}$+Cluster IS is FPT, then $\mathcal{F}$-CF-FVS is FPT. Throughout this section, whenever we refer to a family of graphs, it will refer to a hereditary family of graphs. To prove our result, for a family of graphs $\mathcal{F}$, for which $\mathcal{F}$+Cluster IS is FPT, we will design an FPT algorithm for $\mathcal{F}$-CF-FVS, using the (assumed) FPT algorithm for $\mathcal{F}$+Cluster IS. Our algorithm will use the technique of compression together with branching. We note that the method of iterative compression was first introduced by Reed, Smith, and Vetta [19], and in our algorithm, we (roughly) use only the compression procedure from it.

In the following, we let $\mathcal{F}$ to be a (hereditary) family graphs, for which $\mathcal{F}$+Cluster IS is in FPT. Towards designing an algorithm for $\mathcal{F}$-CF-FVS, we define another problem, which we call $\mathcal{F}$-Disjoint Conflict Free Feedback Vertex Set (to be defined shortly). Firstly, we design an FPT algorithm for $\mathcal{F}$-CF-FVS using an assumed FPT algorithm for $\mathcal{F}$-Disjoint Conflict Free Feedback Vertex Set. Secondly, we give an FPT algorithm for $\mathcal{F}$-Disjoint Conflict Free Feedback Vertex Set using the assumed algorithm for $\mathcal{F}$+Cluster IS. In the following, we formally define the problem $\mathcal{F}$-Disjoint Conflict Free Feedback Vertex Set ($\mathcal{F}$-DCF-FVS, for short)

$\mathcal{F}$-Disjoint Conflict Free Feedback Vertex Set ($\mathcal{F}$-DCF-FVS) **Parameter:** k
Input: A graph G, a graph $H \in \mathcal{F}$, an integer k, a set $W \subseteq V(G)$, a set $R \subseteq V(H) \setminus W$, and a set $\mathcal{C}$, such that the following conditions are satisfied: 1) $V(G) \subseteq V(H)$, 2) $G - W$ is a forest, 3) the number of connected components in $G[W]$ is at most k, and 4) $\mathcal{C}$ is a set of vertex disjoint subsets of $V(H)$.
Question: Is there a set $S \subseteq V(H) \setminus (W \cup R)$ of size at most k, such that $G - S$ is a forest, S is an independent set in H, and for each $C \in \mathcal{C}$, we have $|S \cap C| \neq \emptyset$?

We note that in the definition of $\mathcal{F}$-DCF-FVS, there are two (additional) inputs namely, the set R and the set $\mathcal{C}$. The purpose and need for these sets will become clear when we describe the algorithm for $\mathcal{F}$-DCF-FVS. In Section 4.1, we will prove the following theorem.

▶ **Theorem 7.** *Let $\mathcal{F}$ be a hereditary family of graphs for which there is an* FPT *algorithm for $\mathcal{F}$+*Cluster IS *running in time $f(k)n^{\mathcal{O}(1)}$, where n is the number of vertices in the* *input graph. Then, there is an* FPT *algorithm for* $\mathcal{F}$-DCF-FVS *running in time* $f(k)d^k n^{\mathcal{O}(1)}$*, where* n *is the (total) number of vertices in the input graphs, and* d *is a fixed constant.*

In the rest of the section, we show how we can use the FPT algorithm for $\mathcal{F}$-DCF-FVS to obtain an FPT algorithm for $\mathcal{F}$-CF-FVS.

Algorithm for $\mathcal{F}$-CF-FVS using algorithm for $\mathcal{F}$-DCF-FVS. Let $I = (G, H, k)$ be an instance of $\mathcal{F}$-CF-FVS. We start by checking whether or not G has a feedback vertex set of size at most k, i.e. a set Z of size at most k, such that $G - Z$ is a forest. For this we employ the algorithm for Feedback Vertex Set running in time $\mathcal{O}(3.619^k n^{\mathcal{O}(1)})$ of Kociumaka and Pilipczuk [14]. Here, n is the number of vertices in the input graph. Notice that if G does not have a feedback vertex set of size at most k, then (G, H, k) is a no instance of $\mathcal{F}$-CF-FVS, and we can output a trivial no instance of $\mathcal{F}$-DCF-FVS. Therefore, we assume that (G, k) is a yes instance of Feedback Vertex Set, and let Z be one of its solution. We note that such a set Z can be computed using the algorithm presented in [14]. We generate an instance I_Y of $\mathcal{F}$-DCF-FVS, for each $Y \subseteq Z$, where Y is the guessed (exact) intersection of the set Z with an assumed (hypothetical) solution to $\mathcal{F}$-CF-FVS in I. We now formally describe the construction of I_Y. Consider a set $Y \subseteq Z$, such that Y is an independent set in H. Let $G_Y = G - Y$, $H_Y = H - Y$, $k_Y = k - |Y|$, $W_Y = Z \setminus Y$, $R_Y = (N_H(Y) \setminus W_Y) \cap V(H_Y)$, and $\mathcal{C}_Y = \emptyset$. Furthermore, let $I_Y = (G_Y, H_Y, k_Y, W_Y, R_Y, \mathcal{C}_Y)$, and notice that I_Y is a (valid) instance of $\mathcal{F}$-DCF-FVS. Now we resolve I_Y using the (assumed) FPT algorithm for $\mathcal{F}$-DCF-FVS, for each $Y \subseteq Z$, where Y is an independent set in H. It is easy to see that I is a yes instance of $\mathcal{F}$-CF-FVS if and only if there is an independent set $Y \subseteq Z$ in H, such that I_Y is a yes instance of $\mathcal{F}$-DCF-FVS. From the above discussions, we obtain the following theorem.

▶ **Lemma 8.** *Let* $\mathcal{F}$ *be a family of graphs for which* $\mathcal{F}$-DCF-FVS *admits an* FPT *algorithm running in time* $f(k)n^{\mathcal{O}(1)}$*, where* n *is the (total) number of vertices in the input graph. Then* $\mathcal{F}$-CF-FVS *admits an* FPT *algorithm running in time* $g(k)2^k n^{\mathcal{O}(1)}$*, where* n *is the number of vertices in the input graphs.*

Using Theorem 7 and Lemma 8, we obtain the main theorem of this section.

▶ **Theorem 9.** *Let* $\mathcal{F}$ *be a hereditary family of graphs for which there is an* FPT *algorithm for* $\mathcal{F}$+Cluster IS *running in time* $f(k)n^{\mathcal{O}(1)}$*, where* n *is the number of vertices in the input graph. Then, there is an* FPT *algorithm for* $\mathcal{F}$-CF-FVS *running in time* $f(k)d^k n^{\mathcal{O}(1)}$*, where* n *is the number of vertices in the input graphs of* $\mathcal{F}$-CF-FVS*, and* d *is a fixed constant.*

4.1 FPT Algorithm for $\mathcal{F}$-DCF-FVS

The goal of this section is to prove Theorem 7. Let $\mathcal{F}$ be a (fixed) hereditary family of graphs, for which $\mathcal{F}$+Cluster IS admits an FPT algorithm. We design a branching based FPT algorithm for $\mathcal{F}$-DCF-FVS, using the (assumed) FPT algorithm for $\mathcal{F}$+Cluster IS.

Let $I = (G, H, k, W, R, \mathcal{C})$ be an instance of $\mathcal{F}$-DCF-FVS. In the following we describe some reduction rules, which the algorithm applies exhaustively, in the order in which they are stated.

▶ Reduction Rule 1. Return that $(G, H, k, W, R, \mathcal{C})$ is a no instance of $\mathcal{F}$-DCF-FVS if one of the following conditions are satisfied:

1. if $k < 0$,
2. if $k = 0$ and G has a cycle,
3. $k = 0$ and $\mathcal{C} \neq \emptyset$,

4. $G[W]$ has a cycle;
5. if $|\mathcal{C}| > k$; or
6. there is $C \in \mathcal{C}$, such that $C \subseteq R$.

▶ Reduction Rule 2. If $k = 0$, G is acyclic, and $\mathcal{C} = \emptyset$ then, return that $(G, H, k, W, R, \mathcal{C})$ is a yes instance of $\mathcal{F}$-DCF-FVS.

In the following, we state a lemma, which is useful in resolving those instances where the graph G has no vertices.

▶ **Lemma 10.** *Let $(G, H, k, W, R, \mathcal{C})$ be an instance of $\mathcal{F}$-DCF-FVS, where Reduction Rules 1 is not applicable and $G - W$ has no vertices. Then, in polynomial time, we can generate an instance (G', H', k') of $\mathcal{F}$+Cluster IS, such that $(G, H, k, W, R, \mathcal{C})$ is a yes instance of $\mathcal{F}$-DCF-FVS if and only if (G', H', k') is a yes instance of $\mathcal{F}$+Cluster IS.*

Proof. Let $V_{\mathcal{C}} = (\cup_{C\in\mathcal{C}} C) \setminus R$. We have $V(G') = V(H') = V_{\mathcal{C}}$. For each $C \in \mathcal{C}$, we make $C \setminus R$ a clique in H'. We set $G' = H[V_{\mathcal{C}}]$, and $k' = |\mathcal{C}|$. In the following we show that $(G, H, k, W, R, \mathcal{C})$ is a yes instance of $\mathcal{F}$-DCF-FVS if and only if (G', H', k') is a yes instance of $\mathcal{F}$+Cluster IS.

In the forward direction, let $(G, H, k, W, R, \mathcal{C})$ be a yes instance of $\mathcal{F}$-DCF-FVS, and let S be one of its solution. By construction, S is an independent set in G' and H' of size $\mathcal{C}$.

In the reverse direction, let (G', H', k') be a yes instance of $\mathcal{F}$+Cluster IS, and S be one of its solution. Since Reduction Rule 1 (item 4) is not applicable on $(G, H, k, W, R, \mathcal{C})$, we have $|\mathcal{C}| \leq k$. Therefore, S is of size at most k. By non-applicability of item 6 of Reduction Rule 1, we have $S \cap R = \emptyset$. By construction, $|S \cap C| = 1$, for each $C \in \mathcal{C}$, and S is an independent set in H. From the above discussions, together with the fact that $G = G[W]$ is acyclic, implies that S is a solution to $\mathcal{F}$-DCF-FVS in $(G, H, k, W, R, \mathcal{C})$. This concludes the proof. ◀

Lemma 10 leads us to the following reduction rule.

▶ Reduction Rule 3. If $G - W$ has no vertices, then return the output of algorithm for $\mathcal{F}$+Cluster IS with the instance generated by Lemma 10.

▶ Reduction Rule 4. If there is a vertex $v \in V(G)$ of degree at most one in G, then return $(G - \{v\}, H, k, W \setminus \{v\}, R, \mathcal{C})$.

The safeness of Reduction Rule 4 follows from the fact that a vertex of degree at most one does not participate in any cycle.

▶ Reduction Rule 5. Let $uv \in E(G)$ be an edge of multiplicity greater than 2 in G, and G' be the graph obtained from G by reducing the multiplicity of uv in G to 2. Then, return $(G', H, k, W, R, \mathcal{C})$.

The safeness of Reduction Rule 5 follows from the fact that for an edge, multiplicity of 2 is enough to capture multiplicities of size larger than 2.

▶ Reduction Rule 6. Let $v \in R$ be a degree 2 vertex in G with u and w being its neighbors in G. Furthermore, let G' be the graph obtained from G by deleting v and adding the (multi) edge uw. Then, return $(G', H - \{v\}, k, W, R \setminus \{v\}, \mathcal{C})$.

The safeness of Reduction Rule 6 follows from the fact that a vertex in R cannot be part of any solution and any cycle (in G) containing v must contain both u and w.

▶ Reduction Rule 7. If there is $v \in (V(G) \cap R) \setminus W$, such that v has at least two neighbors in the same connected component of W, then return that $(G, H, k, W, R, \mathcal{C})$ is a no instance of $\mathcal{F}$-DCF-FVS.

▶ Reduction Rule 8. If there is $v \in V(G) \setminus (W \cup R)$, such that v has at least two neighbors in the same connected component of W, then return $(G - \{v\}, H - \{v\}, k-1, W, R \cup N_H(v), \mathcal{C})$.

▶ Reduction Rule 9. Let $v \in V(G) \cap R$, such that $N_G(v) \cap W \neq \emptyset$. Then, return $(G, H, k, W \cup \{v\}, R \setminus \{v\}, \mathcal{C})$.

Let η be the number of connected components in $G[W]$. In the following, we define the measure we use to compute the running time of our algorithm.

$$\mu(I) = \mu((G, H, k, W, R, \mathcal{C})) = k + \eta - |\mathcal{C}|$$

Observe that none of the reduction rules that we described increases the measure, and a reduction rule can be applied only polynomially many time. When none of the reduction rules are applicable, the degree of each vertex in G is at least two, multiplicity of each edge in G is at most two, degree two vertices in G do not belong to the set R, and $G[W]$ and $G[V(G) \setminus W]$ are forests. Furthermore, for each $v \in V(G) \setminus W$, v has at most 1 neighbor (in G) in a connected component of $G[W]$.

In the following, we state the branching rules used by the algorithm. We assume that none of the reduction rules are applicable, and the branching rules are applied in the order in which they are stated. The algorithm will branch on vertices in $V(G) \setminus W$.

▶ Branching Rule 1. If there is $v \in V(G) \setminus W$ that has at least two neighbors (in G), say $w_1, w_2 \in W$. Since Reduction Rule 7 and 8 are not applicable, w_1 and w_2 belong to different connected components of $G[W]$. Also, since Reduction Rule 9 is not applicable, we have $v \notin R$. In this case, we branch as follows.

(i) v belongs to the solution. In this branch, we return $(G - \{v\}, H - \{v\}, k-1, W, R \cup N_H(v), \mathcal{C})$.

(ii) v does not belongs to the solution. In this branch, we return $(G, H, k, W \cup \{v\}, R, \mathcal{C})$.

In one branch when v belongs to the solution, k decreases by 1, and η and $|\mathcal{C}|$ do not change. Hence, μ decreases by 1. In other branch when v is moved to W, number of components in η decreases by at least one, and k and $|\mathcal{C}|$ do not change. Therefore, μ decreases by at least 1. The resulting branching vector for the above branching rule is $(1, 1)$.

If Branching Rule 1 is not applicable, then each $v \in V(G) \setminus W$ has at most one neighbor (in G) in the set W. Moreover, since Reduction Rule 4 is not applicable, each leaf in $G - W$ has a neighbor in W.

In the following, we introduce some notations, which will be used in the description of our branching rules. Recall that $G - W$ is a forest. Consider a connected component T in $G - W$. A path P_{uv} from a vertex u to a vertex v in T is *nice* if u and v are of degree at least 2 in G, all internal vertices (if they exist) of P_{uv} are of degree exactly 2 in G, and v is a leaf in T. In the following, we state an easy proposition, which will be used in the branching rules that we design.

▶ Proposition 1. Let $(G, H, k, W, R, \mathcal{C})$ be an instance of $\mathcal{F}$-DCF-FVS, where none of Reduction Rule 1 to 9 or Branching Rule 1 apply. Then there are vertices $u, v \in V(G) \setminus W$, such that the unique path P_{uv} in $G - W$ is a nice path.

Consider $u, v \in V(G) \setminus W$, for which there is a nice path P_{uv} in T, where T is a connected component of $G - W$. Since Reduction Rule 4 is not applicable, either u has a neighbor in W, or u has degree at least 2 in T. From the above discussions, together with Proposition 1, we design the remaining branching rules used by the algorithm. We note that the branching rules that we describe next is similar to the one given in [3].

▶ Branching Rule 2. Let $v \in V(G) \setminus W$ be a leaf in $G[V(G) \setminus W]$ for which the following holds. There is $u \in V(G) \setminus W$, such that $N_G(u) \cap W \neq \emptyset$ and there is a nice path P_{uv} from u to v in $G[V(G) \setminus W]$. Let $C = V(P_{uv}) \setminus \{u\}$, u' and v' be the neighbors (in G) of u and v in W, respectively. Observe that since Reduction Rule 9 is not applicable, we have $u, v \notin R$. We further consider the following cases, based on whether or not u' and v' are in the same connected component of $G[W]$.

Case 2.A. u' and v' are in the same connected component of $G[W]$. In this case, $G[V(P_{uv}) \cup W]$ contains exactly one cycle, and this cycle contains all vertices of $V(P_{uv})$ (consecutively). Since vertices in W cannot be part of any solution, either u belongs to the solution or a vertex from C belongs to the solution. Moreover, any cycle in G containing v must contain all vertices in $V(P_{uv})$, consecutively. This leads to the following branching rule.

(i) u belongs to the solution. In this branch, we return $(G - \{u\}, H - \{u\}, k - 1, W, R \cup N_H(u), \mathcal{C})$.

(ii) u does not belong to the solution. In this branch, we return $(G - C, H, k, W, R, \mathcal{C} \cup \{C\})$.

In the first branch k decreases by one, and η and $|\mathcal{C}|$ do not change. Therefore, μ decreases by 1. On the second branch $|\mathcal{C}|$ increases by 1, and k and η do not change, and therefore, μ decreases by 1. The resulting branching vector for the above branching rule is $(1, 1)$.

Case 2.B. u' and v' are in different connected component of $G[W]$. In this case, we branch as follows.

(i) u belongs to the solution. In this branch, we return $(G - \{u\}, H - \{u\}, W, k - 1, R \cup N_H(u), \mathcal{C})$.

(ii) A vertex from C is in the solution. In this branch, we return $(G - C, H, k, W, R, \mathcal{C} \cup \{C\})$.

(iii) No vertex in $\{u\} \cup C$ is in the solution. In this branch, we add all vertices in $\{u\} \cup C$ to W. That is, we return $(G, H, k, W \cup (\{u\} \cup C), R \setminus (\{u\} \cup C), \mathcal{C})$.

In the first branch k decreases by one, and η and $|\mathcal{C}|$ do not change. Therefore, μ decreases by 1. On the second branch $|\mathcal{C}|$ increases by 1, and k and η do not change, and therefore, μ decreases by 1. In the third branch, η decreases by one, and k and $|\mathcal{C}|$ do not change. The resulting branching vector for the above branching rule is $(1, 1, 1)$.

▶ Branching Rule 3. There is $u \in V(G) \setminus W$ which has (at least) two nice paths, say P_{uv_1} and P_{uv_2} to leaves v_1 and v_2 (in $G - W$). Let $C_1 = V(P_{uv_1}) \setminus \{u\}$ and $C_2 = V(P_{uv_2}) \setminus \{u\}$. We further consider the following cases depending on whether or not v_1 and v_2 have neighbors (in G) in the same connected component of $G[W]$ and $u \in R$.

Case 3.A. v_1 and v_2 have neighbors (in G) in the same connected component of $G[W]$ and $u \in R$. In this case, $G[W \cup \{u\} \cup C_1 \cup C_2]$ contains (at least) one cycle, and u cannot belong to any solution. Therefore, we branch as follows.

(i) A vertex from C_1 belongs to the solution. In this branch, we return $(G - C_1, H, k, W, R, \mathcal{C} \cup \{C_1\})$.

(ii) A vertex from C_2 belongs to the solution. In this branch, we return $(G - C_2, H, k, W, R, \mathcal{C} \cup \{C_2\})$.

Notice that in both the branches μ decreases by 1, and therefore, the resulting branching vector is $(1, 1)$.

Case 3.B. v_1 and v_2 have neighbors (in G) in the same connected component of $G[W]$ and $u \notin R$. In this case, $G[W \cup \{u\} \cup C_1 \cup C_2]$ contains (at least) one cycle. We branch as follows.

(i) u belongs to the solution. In this branch, we return $(G - \{u\}, H - \{u\}, k - 1, W, R \cup N_H(u), \mathcal{C})$.

(ii) A vertex from C_1 belongs to the solution. In this branch, we return $(G - C_1, H, k, W, R, \mathcal{C} \cup \{C_1\})$.

(iii) A vertex from C_2 belongs to the solution. In this branch, we return $(G-C_2, H, k, W, R, \mathcal{C} \cup \{C_2\})$.

Notice that in all the three branches μ decreases by 1, and therefore, the resulting branching vector is $(1, 1, 1)$.

Case 3.C. If v_1 and v_2 have neighbors in different connected components of $G[W]$ and $u \in R$. In this case, we branch as follows.

(i) A vertex from C_1 belongs to the solution. In this branch, we return $(G-C_1, H, k, W, R, \mathcal{C} \cup \{C_1\})$.

(ii) A vertex from C_2 belongs to the solution. In this branch, we return $(G-C_2, H, k, W, R, \mathcal{C} \cup \{C_2\})$.

(iii) No vertex from $C_1 \cup C_2$ belongs to the solution. In this case, we add all vertices in $\{u\} \cup C_1 \cup C_2$ to W. That is, the resulting instance is $(G, H, k, W \cup (\{u\} \cup C_1 \cup C_2), R \setminus (\{u\} \cup C_1 \cup C_2), \mathcal{C})$.

Notice that in all the three branches μ decreases by 1, and therefore, the resulting branching vector is $(1, 1, 1)$.

Case 3.D. If v_1 and v_2 have neighbors in different connected components of $G[W]$ and $u \notin R$. In this case, we branch as follows.

(i) u belongs to the solution. In this branch, we return $(G - \{u\}, H - \{u\}, k - 1, W, R \cup N_H(u), \mathcal{C})$.

(ii) A vertex from C_1 belongs to the solution. In this branch, we return $(G-C_1, H, k, W, R, \mathcal{C} \cup \{C_1\})$.

(iii) A vertex from C_2 belongs to the solution. In this branch, we return $(G-C_2, H, k, W, R, \mathcal{C} \cup \{C_2\})$.

(iv) No vertex from $\{u\} \cup C_1 \cup C_2$ belongs to the solution. In this case, we add all vertices in $\{u\} \cup C_1 \cup C_2$ to W. That is, the resulting instance is $(G, H, k, W \cup (\{u\} \cup C_1 \cup C_2), R \setminus (\{u\} \cup C_1 \cup C_2), \mathcal{C})$.

Notice that in all the four branches μ decreases by 1, and therefore, the resulting branching vector is $(1, 1, 1, 1)$.

This completes the description of the algorithm. We are now ready to prove Theorem 7.

Proof of Theorem 7. Let $I = (G, H, k, W, R, \mathcal{C})$ be an instance of $\mathcal{F}$-DCF-FVS, and n be the (total) number of vertices in G and H. We prove the correctness of our algorithm by induction on μ.

When $\mu \leq 0$, then Reduction Rule 1 or Reduction Rule 2, correctly resolve the given instance of $\mathcal{F}$-DCF-FVS. This forms the base case of our induction. For the induction hypothesis, we assume that for some $\delta \in \mathbb{N}$, for each $\mu \leq \delta$, the algorithm can correctly resolve the instance. The algorithm either applies one of Reduction Rule 1 to 9 or one of Branching Rule 1 to 3. Proposition 10 implies that either one of Reduction Rule 1 to 9 or Branching Rule 1 is applicable, or one of Branching Rule 2 to 3 is applicable. Each of the reduction rules are safe, they do not increase the measure, and can be applied only polynomially many times. Each of our branching rules are exhaustive, and in each of the branches, the measure strictly decreases. If we apply the reduction rules (exhaustively), either we completely resolve the instance correctly, or eventually apply a branching rule (in polynomial number of application of reduction rules). If one of the branching rules apply, then the measure strictly decreases, and then the induction hypothesis implies the correctness of the algorithm. This concludes the proof of correctness of the algorithm.

In the following, we prove the claimed running time bound for the algorithm for $\mathcal{F}$-DCF-FVS. We note that the worst case branching vector is $(1, 1, 1, 1)$ (Branching Rule 3.D). And, whenever the measure drops below zero, we immediately resolve the instance using one of our reduction rules in time bounded by $f(k) \cdot n^{\mathcal{O}(1)}$. The time required to execute any of the reduction rules is bounded by $f(k) \cdot n^{\mathcal{O}(1)}$. From the above discussions, the running time of our algorithm is bounded by the following expression.

$$T(\mu, n) \leq 4T(\mu - 1, n) + f(\mu)n^{\mathcal{O}(1)}$$

From the above expression, we obtain that the running time of our algorithm is bounded by $\mathcal{O}(4^k f(k) \cdot n^{\mathcal{O}(1)})$. This concludes the proof. ◀

5 FPT Algorithm for $K_{i,j}$-free+Cluster IS

In this section, we give an FPT algorithm for $K_{i,j}$-free+Cluster IS, which is the $\mathcal{F}$+Cluster IS where $\mathcal{F}$ is family of $K_{i,j}$-free graphs. Here, $i, j \in \mathbb{N}$, $1 \leq i \leq j$. In the following we consider a (fixed) family of $K_{i,j}$-free graphs. To design an FPT algorithm for $\mathcal{F}$+Cluster IS, we define another problem called Large $K_{i,j}$-free+Cluster IS. The problem Large $K_{i,j}$-free+Cluster IS is formally defined below.

> Large $K_{i,j}$-free+Cluster IS **Parameter:** k
> **Input:** A $K_{i,j}$-free graph G, a cluster graph H (G and H are on the same vertex set), and an integer k, such that the following conditions are satisfied: 1) H has exactly k connected components, and 2) each connected component of H has at least k^k vertices.
> **Question:** Is there a set $S \subseteq V(G)$ of size k such that S is an independent set in both G and in H?

In Section 5.1, we design a polynomial time algorithm for the problem Large $K_{i,j}$-free+Cluster IS. In the rest of this section, we show how to use the polynomial time algorithm for Large $K_{i,j}$-free+Cluster IS to obtain an FPT algorithm for $K_{i,j}$-free+Cluster IS.

▶ **Theorem 11.** *$K_{i,j}$-free+Cluster IS admits an FPT algorithm running time $\mathcal{O}(k^{k^2} n^{\mathcal{O}(1)})$, where n is the number of vertices is the input graph.*

Proof. Let (G, H, k) be an instance of $K_{i,j}$-free+Cluster IS, and let $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ be the set of connected components in H. If $k \leq 0$, we can correctly resolve the instance in polynomial time (by appropriately outputting yes or no answer). Therefore, we assume $k \geq 1$. If for each $C \in \mathcal{C}$, we have $|V(C)| \geq k^k$, then (G, H, k) is also an instance of Large $K_{i,j}$-free+Cluster IS, and therefore we resolve it in polynomial time using the algorithm for Large $K_{i,j}$-free+Cluster IS (Section 5.1). Otherwise, there is $C \in \mathcal{C}$, such that $|V(C)| < k^k$. Any solution to $K_{i,j}$-free+Cluster IS in (G, H, k) must contain exactly one vertex from C. Moreover, if a vertex $v \in V(C)$ is in the solution, then none of its neighbors in G and in H can belong to the solution. Therefore, we branch on vertices in C as follows. For each $v \in V(C)$, create an instance $I_v(G - (N_H(v) \cup N_G(v)), H - (N_H(v) \cup N_G(v)), k - 1)$ of $K_{i,j}$-free+Cluster IS. If number of connected components in $H - N[C]$ is less than $k - 1$, then we call such an instance I_v as *invalid* instance, otherwise the instance is a *valid* instance. Notice that for $v \in V(C)$, if I_v is an invalid instance, then v cannot belong to any solution. Thus, we branch on valid instances of I_v, for $v \in V(C)$. Observe that (G, H, k) is a yes instance of $K_{i,j}$-free+Cluster IS if and only if there is a valid instance I_v, for $v \in V(C)$, which is a yes instance of $K_{i,j}$-free+Cluster IS. Therefore, we output the OR of results obtained by resolving valid instances I_v, for $v \in V(C)$.

In the above we have designed a recursive algorithm for the problem $K_{i,j}$-free+Cluster IS. In the following, we prove the correctness and claimed running time bound of the algorithm. We show this by induction on the measure $\mu = k$. For $\mu \leq 0$, the algorithm correctly resolve the instance in polynomial time. This forms the base case of our induction hypothesis. We assume that the algorithm correctly resolve the instance for each $\mu \leq \delta$, for some $\delta \in \mathbb{N}$. Next, we show that the correctness of the algorithm for $\mu = \delta + 1$. We assume that $k > 0$, otherwise, the algorithm correctly outputs the answer. The algorithm either correctly resolves the instance in polynomial time using the algorithm for Large $K_{i,j}$-free+Cluster IS, or applies the branching step. When the algorithm resolves the instance in polynomial time using the algorithm for Large $K_{i,j}$-free+Cluster IS, then the correctness of the algorithm follows from the correctness of the algorithm for Large $K_{i,j}$-free+Cluster IS. Otherwise, the algorithm applies the branching step. The branching is exhaustive, and the measure strictly decreases in each of the branches. Therefore, the correctness of the algorithm follows form the induction hypothesis. This completes the proof of correctness of the algorithm.

For the proof of claimed running time notice that the the worst case branching vector is is given by the k^k vector of all 1s, and at the leaves we resolve the instances in polynomial time. Thus, the claimed bound on the running time of the algorithm follows. ◀

5.1 Polynomial Time Algorithm for Large $K_{i,j}$-free+Cluster IS

Consider a (fixed) family of $K_{i,j}$-free graphs, where $1 \leq i \leq j$. The goal of this section is to design a polynomial time algorithm for Large $K_{i,j}$-free+Cluster IS. Let (G, H, k) be an instance of Large $K_{i,j}$-free+Cluster IS, where G is a $K_{i,j}$-free graph and H is a cluster graph with k connected components. We assume that $k > \max\{i + j, 5\}$, as otherwise, we can resolve the instance in polynomial time (using brute-force). Let $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ be the set of connected components in H, such that $|V(C_1)| \geq |V(C_2)| \geq \cdots |V(C_k)|$.

We start by stating/proving some lemmata, which will be helpful is designing the algorithm.

▶ **Lemma 12.** [4] *The number of edges in a $K_{i,j}$-free graph are bounded by $n^{2-\epsilon}$, where $\epsilon \in (0, 1]$.*

▶ **Lemma 13.** *Let (G, H, k) be an instance of* Large $K_{i,j}$*-free*+Cluster IS*. There exists $v \in V(C_1)$, such that for each $C \in \mathcal{C} \setminus \{C_1\}$, we have $|N_G(v) \cap C| \leq \frac{2j|C'|}{k}$.*

Proof. Consider a connected component $C \in \mathcal{C} \setminus \{C_1\}$, and let $x = |C_1|$ and $y = |C|$. Furthermore, let $E(C_1, C) = \{uv \in E(G) \mid u \in C_1, v \in C\}$. In the following, we prove some claims which will be used to obtain the proof of the lemma.

▶ Claim 14. $|E(C_1, C)| \leq jy^i + jx$.

Proof. Consider the partition of $V(C_1)$ in two parts, namely, C_h^1 and C_ℓ^1, where $C_h^1 = \{v \in V(C_1) \mid |N_G(v) \cap V(C)| \geq i\}$ and $C_\ell^1 = V(C_1) \setminus C_h^1$.

$$|E(C_1, C_k)| = \sum_{v \in C_1} |N_G(v) \cap V(C)| = \sum_{v \in C_h^1} |N_G(v) \cap V(C)| + \sum_{v \in C_h^1} |N_G(v) \cap V(C)|.$$

By construction of C_ℓ^1, we have $\sum_{v \in C_\ell^1} |N_G(v) \cap V(C)| < ix$. In the following, we bound $\sum_{v \in C_h^1} |N_G(v) \cap V(C)|$. Since G is a $K_{i,j}$-free graph, therefore, any set of i vertices in $V(C)$ can have at most $j - 1$ common neighbors (in G) from $V(C_1)$, and in particular from C_h^1. Moreover, every $v \in C_h^1$ has at least i neighbors in $N_G(v) \cap V(C)$. Therefore,

$\sum_{v \in C_h^1} |N_G(v) \cap V(C)| \leq i(j-1)\binom{y}{i}$. Hence, $|E(C_1, C_k)| \leq i(j-1)\binom{y}{i} + ix \leq i(j-1)\frac{y^i}{i!} + ix \leq jy^i + jx$

◀

Let $\mathsf{Adeg}_{\mathsf{C_1,C_k}}$ denote average degree of vertices in set C_1 into set C_k. Formally, $\mathsf{Adeg}_{\mathsf{C_1,C_k}} = \frac{|\mathsf{E(C_1,C_k)}|}{|\mathsf{C_1}|}$. We give following upper bound on $\mathsf{Adeg}_{\mathsf{C_1,C_k}}$.

▶ Claim 15. $\mathsf{Adeg}_{C_1,C_k} \leq \frac{2jy}{k^2}$

Proof. Since $|E(C_1, C_k)| \leq jy^i + jx$,

$$\mathsf{Adeg}_{C_1,C_k} \leq j + \frac{jy^i}{x} \tag{1}$$

Using Lemma 12, we give the following bound on Adeg_{C_1,C_k}.

$$\mathsf{Adeg}_{C_1,C_k} \leq \frac{(x+y)^{2-\epsilon}}{x} \leq 4x^{1-\epsilon} \tag{2}$$

To prove the lemma, let us assume the following cases:
Case 1: $x \geq k^2 y^{i-1}$
By substituting x in (1) we get the following bound:

$$\mathsf{Adeg}_{C_1,C_k} \leq j + \frac{jy}{k^2}$$

Since $y > k^2$,

$$\mathsf{Adeg}_{C_1,C_k} \leq \frac{2jy}{k^2}$$

Case 2: $x < k^2 y^{i-1}$
By substituting x in (2) we get the following bound:

$$\mathsf{Adeg}_{C_1,C_k} < 4k^{2(1-\epsilon)} y^{(i-1)(1-\epsilon)} < \frac{4k^2 y}{y^{(2-i)+\epsilon(i-1)}}$$

Since $y \geq k^k$, $y^{(2-i)+\epsilon(i-1)} > \frac{2k^4}{j}$, thus we have following equation:

$$\mathsf{Adeg}_{C_1,C_k} < \frac{2jy}{k^2}$$

◀

Let $deg_{C_k}(v_{C_1})$ denote degree of a vertex $v \in C_1$ in C_k. Since $\mathsf{Adeg}_{C_1,C_k} \leq \frac{2jy}{k^2}$, using Markov's Inequality we get following upper bound on the probability that degree of a vertex $v \in C_1$ in C_k is greater than or equal to $\frac{2jy}{k}$.

$$P\Big(deg_{C_k}(v_{C_1}) \geq \frac{2jy}{k}\Big) \leq \frac{1}{k}$$

Similarly, we can prove that the probability that degree of a vertex $v \in C_1$ in any $C_p \in \mathcal{C}$, $p \in \{2, 3, \ldots, k\}$ is greater than or equal to $\frac{2j|C_p|}{k}$ is at most $\frac{1}{k}$. Using Boole's inequality, we get following upper bound on the probability that degree of a vertex $v \in C_1$ is greater than or equal to $\frac{2j|C_p|}{k}$ for at least one C_p.

$$P\Big(\bigcup_{p \in \{2,3,\cdots,k\}} deg_{C_p}(v_{C_1}) \geq \frac{2j|C_p|}{k}\Big) \leq \frac{1}{k}.(k-1) < 1$$

This implies that probability that degree of a vertex $v \in C_1$ is less than $\frac{2j|C_p|}{k}$ in all $C_p \in \mathcal{C}$ is greater than 0. This completes the proof. ◀ ◀

We are now ready to describe our algorithm. Let (G, H, k) be an instance of $K_{i,j}$-free+Cluster IS such that size of all connected components in H is at least k^k then we use Algorithm 1 to find independent set $S \subseteq V(G)$ such that S contains exactly one vertex from each connected component in H. Let S be initially $\emptyset$.

Algorithm 1 Greedy algorithm for $\mathcal{F}$+Cluster IS when size of all clusters is at least k^k

1: $i = k$
2: **while** $i > 0$ **do**
3: Let $C_1, \cdots, C_i$ be clusters sorted in decreasing order of their size.
4: Delete a vertex v in C_1 which satisfy condition in Lemma 13 and add to solution S, decrease i. Delete C_1 and $N(v)$.
5: **end while**

▶ **Lemma 16.** *Algorithm 1 finds solution for $\mathcal{F}$+Cluster IS where $\mathcal{F}$ is the family of $K_{i,j}$ free graphs and size of each cluster is at least k^k in polynomial time.*

Proof. We first prove the correctness of algorithm using induction on the number of clusters in the graph.
Base case: $t = 1$. We have exactly one connected component in $\mathcal{C}$ say, C_1. Since $k \geq 1$, $|C_1| \geq 1$, we can pick a vertex in C_1 which gives an independent set of G.
Inductive step: Let us assume that the algorithm returns correct solution for $t \leq d - 1$.
Induction: $t = d$. Let $C_1, \cdots, C_d$ be set of connected components sorted in decreasing order. By Lemma 13, there exists a vertex $v \in C_1$ such that degree of v in any C_p, $p \in \{2, 3, \cdots, d\}$, is at most $\frac{2j|C_p|}{d}$. We delete such vertex v, C_1 and $N(v)$ from each $C_p \in \mathcal{C}$. Observe that from each C_p, $p \in \{2, 3, \cdots, d\}$ we have deleted at most $\frac{2j|C_p|}{d}$ vertices, which are neighbors of v. Let C'_p be the cluster after deleting neighbors of v from C_p. It is enough to show that $|C_p|' \geq (d-1)^{(d-1)}$.

$$|C_p|' \geq |C_p| - \frac{2j|C_p|}{d}$$

Without loss of generality, let us assume that $d > 2j$, else we have a polynomial time algorithm that runs in time $\mathcal{O}(n^{2j})$. Hence,

$$\begin{aligned}
|C_p|' &\geq |C_p|\Big(1 - \frac{2j}{d}\Big) \\
&\geq d^d\Big(1 - \frac{2j}{d}\Big) \\
&\geq d^{d-1}(d - 2j) \\
&\geq (d-1)^{(d-1)}
\end{aligned}$$

This proves the correctness of algorithm.
In the algorithm, at each step we either sort the components on the basis of their size or find a vertex of lower degree which can be carried out in polynomial time. Since, the algorithm terminates after at most k iterations, $K_{i,j}$-free+Cluster IS can be solved in polynomial time when size of each cluster is at least k^k. ◀ ◀

References

1. Akanksha Agrawal, Sushmita Gupta, Saket Saurabh, and Roohani Sharma. Improved algorithms and combinatorial bounds for independent feedback vertex set. In *IPEC*, volume 63 of *LIPIcs*, pages 2:1–2:14, 2016.
2. Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In *LATIN*, volume 9644 of *LNCS*, pages 1–13. Springer, 2016.
3. Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. Simultaneous feedback vertex set: A parameterized perspective. In *STACS*, pages 7:1–7:15, 2016.
4. Béla Bollobás. *Extremal graph theory*. Courier Corporation, 2004.
5. Leizhen Cai and Junjie Ye. Dual connectedness of edge-bicolored graphs and beyond. 8635:141–152, 2014.
6. Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188 – 1198, 2008.
7. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
8. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
9. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
10. Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009.
11. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.
12. Zoltán Füredi. On the number of edges of quadrilateral-free graphs. *Journal of Combinatorial Theory, Series B*, 68(1):1–6, 1996.
13. Pallavi Jain, Lawqueen Kanesh, and Pranabendu Misra. Conflict free version of covering problems on graphs: Classical and parameterized. *CSR(to appear)*, 2018.
14. Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.
15. John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.
16. Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On parameterized independent feedback vertex set. *Theoretical Computer Science*, 461:65–75, 2012.
17. Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. Fpt algorithms for connected feedback vertex set. *Journal of Combinatorial Optimization*, 24(2):131–146, 2012.
18. Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.
19. Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
20. Jan Arne Telle and Yngve Villanger. FPT algorithms for domination in biclique-free graphs. In *Algorithms - ESA*, pages 802–812, 2012.
21. René van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015.