On Parameterized Independent Feedback Vertex Set

Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh

The Institute of Mathematical Sciences, Chennai, India.
{neeldhara|gphilip|vraman|saket}@imsc.res.in

Abstract. We investigate a generalization of the classical Feedback Vertex Set (FVS) problem from the point of view of parameterized algorithms. Independent Feedback Vertex Set (IFVS) is the "independent" variant of the FVS problem and is defined as follows: given a graph G and an integer k, decide whether there exists $F \subseteq V(G)$, $|F| \leq k$, such that $G[V(G) \setminus F]$ is a forest and $G[F]$ is an independent set; the parameter is k. Note that the similarly parameterized versions of the FVS problem — where there is no restriction on the graph $G[F]$ — and its connected variant CFVS — where $G[F]$ is required to be connected — have been extensively studied in the literature. The FVS problem easily reduces to the IFVS problem in a manner that preserves the solution size, and so any algorithmic result for IFVS directly carries over to FVS. We show that IFVS can be solved in time $O(5^k n^{O(1)})$ time where n is the number of vertices in the input graph G, and obtain a cubic ($O(k^3)$) kernel for the problem. Note the contrast with the CFVS problem, which does not admit a polynomial kernel unless $CoNP \subseteq NP/Poly$.

1 Introduction

Feedback Vertex Set(FVS) is a classical NP-complete problem and has been extensively studied in all subfields of algorithms and complexity. In this problem we are given an undirected graph G and a positive integer k as input, and the goal is to check whether there exists a subset $F \subseteq V(G)$ of size at most k such that $G[V(G) \setminus F]$ is a forest. This problem originated in combinatorial circuit design and found its way into diverse applications such as deadlock prevention in operating systems, constraint satisfaction and Bayesian inference in artificial intelligence. We refer to the survey by Festa, Pardalos and Resende [11] for further details on the algorithmic study of feedback set problems in a variety of areas like approximation algorithms, linear programming and polyhedral combinatorics.

In this paper we introduce a variant of FVS, namely, Independent Feedback Vertex Set(IFVS) and study it in the realm of parameterized complexity. In IFVS, given a graph G and a positive integer k, the objective is to check whether there exists a vertex-subset F of size at most k such that $G[V(G) \setminus F]$ is a forest and $G[F]$ is an independent set.

Parameterized complexity is a two-dimensional generalization of "P vs. NP" where, in addition to the overall input size n, one studies how a secondary

measurement that captures additional relevant information affects the computational complexity of the problem in question. Parameterized decision problems are defined by specifying the input, the parameter and the question to be answered. The two-dimensional analogue of the class P is decidability within a time bound of $f(k)n^c$, where n is the total input size, k is the parameter, f is some computable function and c is a constant that does not depend on k or n. A parameterized problem that can be decided in such a time-bound is termed *fixed-parameter tractable* (FPT). For general background on the theory see the textbooks by Downey and Fellows [8], Flum and Grohe [12] and Niedermeier [23].

A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm (the degree of polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance down to an instance with size bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. Kernelization has been at the forefront of research lately and many new results have appeared; see the surveys by Guo and Niedermeier [17] and Bodlaender [3]. Some of the most significant recent results are meta theorems for kernelization [4, 15, 19], use of probabilistic tools and Fourier analysis [1, 18] and non-trivial applications of combinatorial min-max results [10, 13, 26].

FVS has been extensively studied in parameterized algorithms. The earliest known FPT-algorithms for FVS go back to the late 80's and the early 90's [2, 9] and used the seminal Graph Minor Theory of Robertson and Seymour. Subsequently, several algorithms for FVS with running times of the form $O(2^{O(k)}n^{O(1)})$ were designed using a technique known as iterative compression. After several rounds of improvement, the current best FPT-algorithm for FVS runs in time $O(3.83^k kn^2)$ [5].

Our motivation for studying the independent variant of FVS is three-fold:

- Somewhat surprisingly, the independent variant of FVS has not been considered in the literature until now. This is in stark contrast to the fact that the independent variants of other problems like Dominating Set — Independent Dominating Set [14, 16, 20] — and Odd Cycle Transversal — Independent Odd Cycle Transversal [24, 21] — have been extensively investigated.
- A simple polynomial time parameter preserving reduction — subdivide every edge once — shows that IFVS is a more general problem than FVS. So a fast FPT algorithm for IFVS directly implies an FPT algorithm for FVS which runs as fast, except for an additive polynomial factor for the transformation.
- FVS admits an $O(k^2)$ kernel [26], while its connected variant CFVS does not admit a kernel of any polynomial size (under certain complexity-theoretic assumptions) [22]. Our final motivation for studying IFVS was to find whether it has a polynomial kernel like FVS, or no polynomial kernel (under the same assumptions) like CFVS.

A formal statement of the parameterized Independent Feedback Vertex Set problem is as follows:

Independent Feedback Vertex Set(IFVS)

Input:	A graph $G = (V, E)$, and an integer k.
Parameter:	k
Question:	Does there exist a subset S of at most k vertices such that $G[S]$ is an independent set, and $G \setminus S$ induces a forest?

Our results. We obtain an FPT algorithm which solves IFVS in time $O(5^k n^{O(1)})$, more succinctly represented as $O^\star(5^k)$ where the $O^\star$ notation hides polynomial factors in the running time. This is as fast as the previous best algorithm for FVS which runs in $O^\star(5^k)$ time [6] (The current fastest algorithm for FVS runs in $O^\star(3.83^k)$ time [5].). Our second result is a polynomial kernel for IFVS; we obtain a kernel of size $O(k^3)$ for the problem. This is in contrast to the fact that Connected Feedback Vertex Set does not admit a polynomial kernel unless $CoNP \subseteq NP/Poly$ [22]. Our kernelization procedure makes use of the q-expansion lemma, a generalization of Hall's Theorem, with q set to $k + 2$.

While the overall idea of our algorithm follows the established paradigm of solving FVS using *iterated compression*, we come up against some subtle differences. For instance, since we need to guarantee that the solution obtained is an independent set, we cannot, unlike with FVS, preprocess of our graph so that its minimum degree is 3. A transformation of the input graph to a graph with minimum degree 3 has played a crucial role in all previous algorithms for FVS. We overcome this handicap by devising a new measure and using an interesting polynomial time subroutine to obtain the FPT algorithm for IFVS.

2 An $O^\star(5^k)$ Algorithm

We now describe an algorithm that solves the IFVS problem in $O^\star(5^k)$ time. The algorithm starts by exhaustively applying two reduction rules which get rid of vertices of small degree and consecutive vertices of degree two in the graph. Since a vertex of degree zero or one does not form part of any cycle, no minimal feedback vertex set contains such a vertex. This justifies the following rule:

Reduction Rule 1 *Delete all vertices of degree at most one in the input graph.*

In contrast to plain FVS — see, for example, the quadratic kernel argument due to Thomassé [26] — the independence requirement of IFVS prevents us from freely bypassing every vertex of degree exactly two. However, it *is* safe to delete all but one of a sequence of two or more consecutive vertices of degree two:

Reduction Rule 2 *Let x, y be two adjacent vertices of degree exactly two in the input graph G, and a, b be the other neighbors of x, y, respectively. Delete the vertex x and add the edge $\{a, y\}$, as in Figure 1.*

Out of two adjacent vertices of degree two, at most one need be in any minimal IFVS. This implies that Reduction Rule 2 is safe.

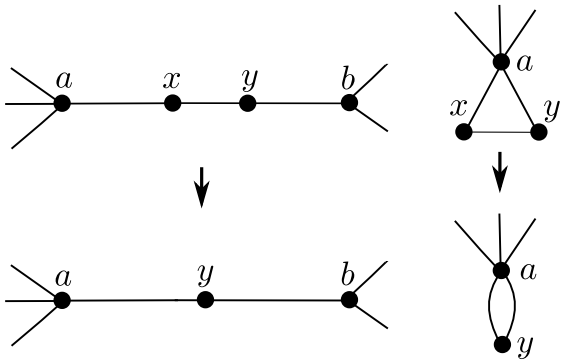

Fig. 1. Bypassing a degree-two vertex which has another such vertex adjacent to it.

Claim 1 [⋆][1] *Let (G, k) be an input instance of* IFVS*, and let x, y, a, b be as in Reduction Rule 2. Let G' be the graph obtained by applying the rule to G. Then (G, k) is a* YES *instance of* IFVS *if and only if (G', k) is a* YES *instance of* IFVS.

The algorithm applies these reduction rules exhaustively; in the following, we assume that the input graph G is reduced with respect to both these rules.

The algorithm now checks in $O^\star(3.83^k)$ time whether G has an FVS of size at most k, by invoking as a subroutine the algorithm due to Chen et al. [5]. If the subroutine returns NO, then G does not have an IFVS of size at most k either, and so the algorithm returns NO. Otherwise, let F be an FVS of G of size at most k returned by the subroutine. The algorithm now passes G, F to a *search routine*, described below, which either says (correctly) that G has no IFVS of size at most k, or returns an IFVS X of G of size at most k.

We now describe the search routine. The input to the search routine is a pair $(G, F \subseteq V(G))$ where F is an FVS of G of size at most k. The goal of the search routine is to output an IFVS X of G of size at most k, if it exists, or to report that no such IFVS exists. The search routine guesses the set $Y = X \cap F; 0 \leq |Y| \leq k$. For this, the routine tries each subset $Y \subseteq F$ of size at most k. If $G[Y]$ is not an independent set, then the routine rejects this guess. Otherwise, let $N = F \setminus Y$. Note that the remaining $k - |Y|$ vertices in X are in $H = V(G) \setminus F$, so that the remaining task is to find an IFVS $Z \subseteq H$ for the subgraph $G[N \cup H]$, such that no vertex in Z is adjacent to any vertex in Y. If $G[N]$ is not a forest, then the routine rejects this guess of Y. Otherwise, it deletes the vertices in Y and tries to find an IFVS $Z \subseteq H$ of the required kind. For this, it first colors red those vertices in H which are adjacent to some vertex in Y, and all the other vertices in H white; red vertices are not to be picked in Z. Note that both $G[N]$ and $G[H]$ are now forests. The routine branches on the vertices in $G[H]$, as described in the three steps in Algorithms 1.

[1] Due to space constraints, proofs of results labeled with a ⋆ have been deferred to a full version of the paper.

We use the following measure to bound the depth of the branching: $\mu = b + c - u$, where:

1. b is the *budget* — the number of additional vertices that can be added to the IFVS being constructed. Initially, $b = k - |Y| \leq k$.
2. c is the number of *components* (trees) in $G[N]$. Initially, $1 \leq c \leq k$.
3. u is the number of *useful* vertices in H. We say that a vertex in H is useful if it is not red, has degree exactly two in G, and both its neighbors are in N.

Algorithm 1 BRANCH(G, H, N), Step 1. See the main text for details.

```
1: if a vertex v in H has at least two neighbors in N and total degree at least three then
2:     if v has two neighbors in the same tree in N then  ▷ v must be picked in any solution
3:         if v is red then
4:             Stop and return NO.
5:         else
6:             pick v to be in the solution.
7:     else
8:         if v is red then
9:             Move v from H to N.
10:        else
11:            Branch on v.
```

If a vertex v in H has two neighbors in any tree in N, then any FVS which is contained in H must contain v. Therefore, if at any point during the branching, there is a red vertex which has two neighbors in any tree in N, then the routine stops and returns NO as the answer. Further, if at any point the budget b or the measure μ becomes negative, the routine stops and returns NO as the answer; this is justified by Claim 3.

"Picking a vertex v in H to be in the solution" consists of coloring all its white neighbors in H red, deleting v from the graph, reducing b by one, and applying Reduction Rules 1 and 2 to the resulting graph. Observe that the arguments for the correctness for Reduction Rules 1 and 2 go through even if one or more of the vertices involved are colored red (and therefore not available for selection into the IFVS being built). "Picking a vertex v in H to *not* be in the solution" consists of moving v from H to N.

"Branching on a vertex v" consists of the following: First pick v in the solution and recurse on the remaining problem. If this returns an FVS X of G of size at most k, then return X and stop. Otherwise, pick v to be *not* in the solution, recurse on the remaining problem, and return the answer.

If none of the branches applies, then by Claim 2 below, every vertex in H has degree exactly two, and both its neighbors are in N. It is now sufficient for the algorithm to find a smallest set $W \subseteq H$ of white vertices that forms an

Algorithm 1 BRANCH(G, H, N), Step 2

12: **if** a vertex v in H is a leaf in $G[H]$ and its only neighbor w in H has a neighbor in N **then**
13: **if** w is red **then**
14: Move w from H to N.
15: **else**
16: Branch on w.

Algorithm 1 BRANCH(G, H, N), Step 3

17: **if** a vertex v in H has at least two neighbors in H which are leaves in $G[H]$ **then**
18: **if** v is red **then**
19: Move v from H to N.
20: **else**
21: Branch on v.

FVS — note that this set is already independent — of $G[N \cup H]$, if it exists, in polynomial time. For this, the algorithm moves all red vertices of H to N and then applies the polynomial-time algorithm due to Chen et al. [6, Lemma 6] which solves this problem in $O(|V(G)|^2)$ time. If there is no such set W, or if $|W| > k - |Y|$, then the search routine outputs NO; otherwise it outputs $Y \cup W$ as an IFVS of G of size at most k.

Claim 2 [$\star$] *Let G be a graph obtained by the search routine to which none of the branches applies, and let N, H be as in the description of the routine. Then every vertex in H has degree exactly two, and both its neighbors are in N.*

Recall that the search routine returns NO as the answer if, at any point during the branching, the budget b or the measure μ becomes negative. This is justified by the following claim.

Claim 3 [$\star$] *Consider a point where the search routine is applied to the graph G, and either the budget b or the measure μ has become negative. Let Y' be the set of vertices chosen by the algorithm to be in the solution till this point, and let N, H be as in the description of the routine. Then there is no* IFVS *of G of size at most k which contains all the vertices in Y'.*

The correctness of the algorithm follows from the above discussion. Observe that $\mu \leq 2k$ at the start of the branching. We bound the running time by showing that μ decreases by at least one on each branch, to obtain:

Theorem 1. [$\star$] *The Independent Feedback Vertex Set problem can be solved in $O^\star(5^k)$ time.*

3 A Cubic Kernel

In this section we describe a kernelization algorithm which yields a kernel of size $O(k^3)$ for the IFVS problem. Given an instance (G, k) of IFVS, the kernelization

algorithm applies a few reduction rules exhaustively. While applying some of the reduction rules, the algorithm colors certain vertices red to indicate that these vertices are not to be picked in any minimal IFVS of size at most k of the resulting graph. At the end of this process, the algorithm either solves the problem (giving either YES or NO as the answer), or it yields an equivalent vertex-colored instance $(H', \ell); \ell \leq k$ whose size is bounded by $O(k^3)$. If the procedure solves the problem, then the algorithm returns a trivial YES or NO instance, as the case may be, of constant size. Otherwise, as the last step, the algorithm adds a gadget to represent the colors of the vertices to obtain an equivalent uncolored instance $(H, k'); k' \leq k$ of size $O(k^3)$.

For ease of notation we use (G, k) to denote the input to each reduction rule, and (H, k') to denote the output. Note that, in general, (G, k) is not the same as the original input instance, and (H, k') is not the final output instance. We also use the term "non-red IFVS" to denote an IFVS which contains no red vertex.

The kernelization algorithm starts by exhaustively applying Reduction Rules 1 and 2 from the previous section to the input graph. Since the graph has no red vertices yet, every IFVS of the graph at this stage is non-red. The algorithm then exhaustively applies the following reduction rule to the resulting instance:

Reduction Rule 3 *Let $u_1, u_2, \ldots, u_r$ be $r \geq k+2$ vertices of degree exactly two in the graph G such that $N(u_i) = \{a, b\}; 1 \leq i \leq r$. Delete the vertices $u_3, u_4 \ldots, u_r$ and color u_1, u_2 red to obtain H. The resulting instance is (H, k).*

If neither of a, b is in an IFVS I, then at least $k+1$ of the vertices $u_1, u_2, \ldots, u_r$ must be in I, or else a, b and the remaining u_is form a cycle in $G[V(G) \setminus I]$. Thus any solution must contain at least one of a, b, and so must exclude all of the u_is. We cannot safely delete all the u_is outright, because this will get rid of the cycles formed by these vertices "for free". But we can delete all but two of the u_is; these two will "remember" the presence of these cycles. A formal version of this intuition justifies the following claim:

Claim 4 [$\star$] *Let (G, k) be an instance of* IFVS*, and let (H, k) be the instance obtained by applying Reduction Rule 3 to (G, k). Then G has a non-red* IFVS *of size at most k if and only if H has a non-red* IFVS *of size at most k.*

Our next reduction rule rules out the presence of too many vertex-disjoint cycles of a certain kind in the graph:

Reduction Rule 4 *From the graph G, construct an auxiliary graph G' as follows. The vertex set of G' is the same as that of G, and the edge $\{u, v\}$ is present in G' if and only if there are at least two vertices x, y in G such that $N(x) = N(y) = \{u, v\}$. Find the size m of a largest-size matching in G'. If $m \geq k+1$, then return* NO *and stop. Otherwise return (G, k).*

Claim 5 [$\star$] *Reduction Rule 4 is safe.*

We now take care of vertices which lie on many cycles which are otherwise disjoint.

Definition 1. *Let v be a vertex in a graph G, and let $\ell \in \mathbb{N}$. An* ℓ-flower passing through v *is a set of ℓ distinct cycles in G such that each cycle contains v and no two cycles share any vertex other than v. The vertex v is said to be at the* center *of the flower.*

Reduction Rule 5 *Let v be a vertex in the graph G which is at the center of a $k+1$-flower. If v is red, then return* NO *and stop. Otherwise, color all neighbors of v red and delete v from G to obtain the graph H; the resulting instance is $(H, k-1)$.*

The correctness of this rule follows essentially from the fact that any vertex which is at the center of a $k+1$-flower must be present in any FVS of the graph of size at most k:

Claim 6 [$\star$] *Let (G, k) be an instance of* IFVS*, and let $(H, k-1)$ be the instance obtained by applying Reduction Rule 5 to (G, k). Then G has a non-red* IFVS *of size at most k if and only if H has a non-red* IFVS *of size at most $k-1$.*

The next reduction rule takes care of boundary conditions; its correctness is self-evident.

Reduction Rule 6
1. *Let v be a vertex in the graph G which has a self-loop. If v is red, then return* NO *and stop. Otherwise, color all neighbors of v red and delete v from G to obtain the graph H; the resulting instance is $(H, k-1)$.*
2. *If $\{x, y\}$ is an edge with multiplicity more than two, then reduce its multiplicity to two.*
3. *If $k = 0$ and G is not a forest, then return* NO *and stop.*
4. *If $k \geq 0$ and G is a forest, then return* YES *and stop.*

We now introduce a reduction rule which helps us bound the maximum degree of any vertex in the graph. To describe the intuition behind this rule, we need two known results.

Firstly, as observed by Thomassé, in a graph reduced with respect to Reduction Rules 5 and 6, we can find, in polynomial time, a small set of vertices that intersects every cycle which passes through any given vertex. More formally,

Theorem 2. [25, Corollary 2.1] *Let v be a vertex of a graph G, and let there be no self-loop at v. If there is no $k+1$-flower passing through v, then there exists a set $X \subseteq V(G) \setminus \{v\}$ of size at most $2k$ which intersects every cycle that passes through v, and such a set can be found in polynomial time.*

For a vertex v in G, we use X_v to denote a set of size at most $2k$ of the kind guaranteed to exist by the theorem.

The second result that we need is an "expansion lemma" which is a generalization of Hall's theorem for bipartite graphs. This was first observed by Thomassé [25]; we use a stricter version due to Fomin et al. [13]. Consider a bipartite graph G with vertex bipartition $A \uplus B$. Given subsets $S \subseteq A$ and $T \subseteq B$,

we say that S has $|S|$ q-stars in T if to every $x \in S$ we can associate a subset $F_x \subseteq N(x) \cap T$ such that (a) for all $x \in S$, $|F_x| = q$; (b) for any pair of vertices $x, y \in S$, $F_x \cap F_y = \emptyset$. Observe that if S has $|S|$ q-stars in T then every vertex x in S could be thought of as the center of a star with its q leaves in T, with all these stars being vertex-disjoint. Further, a collection of $|S|$ q-stars is also a family of q edge-disjoint matchings. The q-Expansion Lemma states a sufficient condition for a special kind of q-star to exist in a bipartite graph:

Lemma 1 ([13]). [The q–Expansion Lemma] *Let q be a positive integer, and let m be the size of the maximum matching in a bipartite graph G with vertex bipartition $A \uplus B$. If $|B| > mq$, and there are no isolated vertices in B, then there exist nonempty vertex sets $S \subseteq A, T \subseteq B$ such that S has $|S|$ q-stars in T and no vertex in T has a neighbor outside S. Furthermore, the sets S, T can be found in time polynomial in the size of G.*

These two results imply that if v is a vertex of sufficiently large degree in a graph reduced with respect to reduction rules Rules 1 to 6, then we can find, in polynomial time, a $k+2$-sized "almost-flower" passing through v. More precisely, we can find a nonempty subset $S \subseteq X_v$ such that for each $s \in S$ there is a set C_s of $k+2$ cycles whose only common vertices are s and v. Further, for any $t \in S, t \neq s$, v is the only vertex shared by cycles in C_s and C_t:

Claim 7 [$\star$] *Let (G, k) be an instance of* IFVS *where G is reduced with respect to Reduction Rules 1 to 6. If G has a vertex v with degree at least $4k + (k+2)2k$, then in polynomial time we can find a set $S \subseteq V(G) \setminus \{v\}$ and a set of components $\mathcal{C}$ of $G \setminus S \cup \{v\}$ such that*

1. *there is exactly one edge in G from v to each component in $\mathcal{C}$,*
2. *each $C \in \mathcal{C}$ induces a tree, and*
3. *there exists a set of at least $(k+2)$ components in $\mathcal{C}$ corresponding to each $s \in S$ such that these sets are pairwise disjoint, and there is an edge from each $s \in S$ to each of its associated components.*

Given a vertex v and a set S as in Claim 7, it can be shown (See the proof of Claim 8) that if F is an FVS of size at most k and $v \notin F$, then $S \subseteq F$. This allows us to reduce the number of edges in the graph in the following way:

Reduction Rule 7 *Let v be a vertex in G, let $S \subseteq V(G) \setminus \{v\}$, and let $\mathcal{C}$ be a set of (not necessarily all) components of $G \setminus S \cup \{v\}$ which satisfy the conditions of Claim 7. Color the neighbors of v in the components in $\mathcal{C}$ red, and delete the edges between v and these newly reddened vertices. For each $s \in S$, if there does not exist a pair a, b of red vertices in G be such that $N(a) = N(b) = \{v, s\}$, then add two new red vertices a, b and the edges $\{v, a\}, \{a, s\}, \{s, b\}, \{b, v\}$. Let the resulting graph be H. The new instance is (H, k).*

Note that the above rule is quite similar to the reduction rule introduced by Thomassé for obtaining a quadratic kernel for FVS [26]. The only difference

here is that we need $k+2$ "private" (in the sense stated in Claim 7) components per vertex in S for the rule to apply, while the FVS reduction rule required only two such components per vertex in S. As shown below, this number contributes a multiplicative factor in the size of the final kernel. Hence our kernel has size $O(k^3)$, while the size of the FVS kernel is quadratic in k. Let F be an FVS of a graph G and let $A \subseteq F$. If $B \subseteq V(G)$ is such that $|B| = |A|$ and B intersects exactly the same set of cycles in G as A does, then $F' = (F \setminus A) \cup B$ is always an FVS of G of size $|F|$. But if F is an IFVS of G, then it is not always true that F' is an IFVS of G. This is precisely the reason for the requirement of $k+2$ components per vertex; these many components are needed before it can be argued that either v or all of S must be in every solution of size at most k. This latter fact is central to the correctness of this reduction rule:

Claim 8 [$\star$] *Let G, k, H be as in the description of Reduction Rule 7. Then G has a non-red* IFVS *of size at most k if and only if H has a non-red* IFVS *of size at most k.*

Each reduction rule can be applied in polynomial time, and each rule which changes the graph decrements the sum of the number of vertices and edges in the graph. Hence all the reduction rules can be exhaustively applied in polynomial time:

Claim 9 [$\star$] *By repeatedly applying Reduction Rules 1 to 7 to an input instance (G, k) of* IFVS*, in polynomial time we can either obtain a* YES *or* NO *answer, or an equivalent instance (H, k') to which none of the rules applies.*

Starting from a YES instance, these reduction rules produce an instance of size $O(k^3)$:

Claim 10 [$\star$] *Let (G, k) be an input instance of* IFVS*, and let (H, k') be a colored graph obtained from (G, k) by exhaustively applying Reduction Rules 1 to 7. If (G, k) is a* YES *instance, then H has at most $k + 16k^2 + 8(k+2)k^2$ vertices and at most $20k^2 + 9(k+2)k^2$ edges.*

This claim justifies the last reduction rule; it is easy to see that the rule can be applied in polynomial time.

Reduction Rule 8 *Let (G, k) be an instance of* IFVS*, and let (H, k') be the instance obtained by exhaustively applying Reduction Rules 1–7. If H has more than $k + 16k^2 + 8(k+2)k^2$ vertices or more than $20k^2 + 9(k+2)k^2$ edges, then return* NO *and stop. Otherwise return (H, k').*

Starting with an instance (G, k), at this point in the kernelization algorithm, we have either obtained a (correct) YES or NO answer, or we have an equivalent colored instance (G', k') where $k' \leq k$ and the size of G is bounded by $O(k^3)$. In former case, the algorithm constructs a trivial YES or NO instance, respectively, and returns it. In the latter case, the algorithm "un-colors" the colored instance to obtain an instance of IFVS with no colors, and returns this instance.

Claim 11 [⋆] *From a colored instance (G', k') produced by the kernelization algorithm, an equivalent uncolored instance $(H, k'+1)$ can be constructed in polynomial time by adding $O(k'^3)$ vertices and edges.*

Putting all these together, we have:

Theorem 3. Independent Feedback Vertex Set *has a kernel of size $\mathcal{O}(k^3)$.*

4 Discussion and Conclusion

In this paper we investigated the parameterized complexity of a generalized version of the well known FVS problem, namely IFVS. We obtained an FPT algorithm which solves the problem in $O^\star(5^k)$ time, and a polynomial kernel of size $O(k^3)$. This work adds to the study of the variants of parameterized FVS, which has yielded some interesting contrasts so far. Plain FVS, without any constraints, is FPT [5] and has a quadratic kernel [26]. The connected variant, CFVS, is FPT but does not admit any polynomial kernel unless $CoNP \subseteq NP/Poly$ [22]. Adding to this picture, we show in this paper that the independent variant is FPT and admits a cubic kernel.

A natural next question to ask is whether the *directed* versions of these problems are FPT. It is known that Directed Feedback Vertex Set (DFVS) is FPT [7]. In contrast, it turns out that Independent Directed Feedback Vertex Set (IDFVS) is unlikely to be FPT:

Theorem 4. [⋆] *Given a directed graph G and a positive integer parameter k, it is W[1]-hard to find if there is a set S of at most k vertices in G such that (i) $G[V(G) \setminus S]$ has no directed cycle and (ii) $G[S]$ is an independent set.*

We leave open the parameterized complexity of *Connected* DFVS.

References

1. Alon, N., Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: Solving max-r-sat above a tight lower bound. In: SODA. pp. 511–517 (2010)
2. Bodlaender, H.L.: On disjoint cycles. In: Schmidt, G., Berghammer, R. (eds.) Proceedings on Graph–Theoretic Concepts in Computer Science (WG '91). LNCS, vol. 570, pp. 230–238. Springer (1992)
3. Bodlaender, H.L.: Kernelization: New Upper and Lower Bound Techniques. In: Chen, J., Fomin, F.V. (eds.) Revised Selected Papers of IWPEC 2009. LNCS, vol. 5917, pp. 17–37. Springer (2009)
4. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (meta) kernelization. In: FOCS. pp. 629–638 (2009)
5. Cao, Y., Chen, J., 0002, Y.L.: On feedback vertex set new measure and new structures. In: SWAT. Lecture Notes in Computer Science, vol. 6139, pp. 93–104 (2010)
6. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved Algorithms for Feedback Vertex Set Problems. Journal of Computer and System Sciences 74(7), 1188–1198 (2008)

7. Chen, J., Liu, Y., Lu, S., O'sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. Journal of the ACM 55(5), 21:1–21:19 (2008)
8. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer-Verlag, New York (1999)
9. Downey, R.G., Fellows, M.R.: Fixed parameter tractability and completeness. In: Complexity Theory: Current Research. pp. 191–225. Cambridge University Press (1992)
10. Fellows, M.R., Guo, J., Moser, H., Niedermeier, R.: A generalization of nemhauser and trotter's local optimization theorem. In: STACS. pp. 409–420 (2009)
11. Festa, P., Pardalos, P.M., Resende, M.G.: Feedback set problems. In: Handbook of Combinatorial Optimization. pp. 209–258. Kluwer Academic Publishers (1999)
12. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin (2006)
13. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and Kernelization. In: Proc. of the 28th Symposium on Theoretical Aspects of Computer Science (STACS) (2011), to appear, available at `http://arxiv.org/abs/1010.1365`
14. Fomin, F.V., Grandoni, F., Kratsch, D.: Solving connected dominating set faster than 2^n. Algorithmica 52(2), 153–166 (2008)
15. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: SODA. pp. 503–510 (2010)
16. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica 20(4), 374–387 (1998)
17. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. SIGACT News 38(1), 31–45 (2007)
18. Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: A probabilistic approach to problems parameterized above or below tight bounds. J. Comput. Syst. Sci. 77(2), 422–429 (2011)
19. Kratsch, S.: Polynomial kernelizations for MIN $\mathrm{F}^+ \mathrm{Pi}_1$ and MAX NP. In: STACS. pp. 601–612 (2009)
20. Lokshtanov, D., Mnich, M., Saurabh, S.: Linear kernel for planar connected dominating set. In: TAMC. Lecture Notes in Computer Science, vol. 5532, pp. 281–290 (2009)
21. Marx, D., O'Sullivan, B., Razgon, I.: Treewidth reduction for constrained separation and bipartization problems. In: STACS. pp. 561–572 (2010)
22. Misra, N., Philip, G., Raman, V., Saurabh, S., Sikdar, S.: Fpt algorithms for connected feedback vertex set. In: WALCOM. Lecture Notes in Computer Science, vol. 5942, pp. 269–280 (2010)
23. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms, Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, Oxford (2006)
24. Reed, B.A., Smith, K., Vetta, A.: Finding odd cycle transversals. Oper. Res. Lett. 32(4), 299–301 (2004)
25. Thomassé, S.: A quadratic kernel for feedback vertex set. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009). pp. 115–119. Society for Industrial and Applied Mathematics (2009)
26. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Transactions on Algorithms 6, 32:1–32:8 (2010)