On Parameterized Independent Feedback Vertex Set

Neeldhara Misra[a], Geevarghese Philip[a,*], Venkatesh Raman[a], Saket Saurabh[a]

[a]*The Institute of Mathematical Sciences, Chennai, India. 600 113*

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 version of the FVS problem — where there is no restriction on the graph $G[F]$ — has been extensively studied in the literature. The connected variant CFVS — where $G[F]$ is required to be connected — has received some attention as well. 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 $CoNP \subseteq NP/Poly$.

Keywords: Fixed-Parameter Tractable Algorithms, Polynomial Kernels, Feedback Vertex Set Problems

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

*Corresponding author.

Email addresses: neeldhara@imsc.res.in (Neeldhara Misra), gphilip@imsc.res.in (Geevarghese Philip), vraman@imsc.res.in (Venkatesh Raman), saket@imsc.res.in (Saket Saurabh)

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 [1] 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 [2], Flum and Grohe [3] and Niedermeier [4].

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 [5] and Bodlaender [6].

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 [7, 8] 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 deterministic FPT-algorithm for FVS runs in

time $O(3.83^k kn^2)$ [9]. The fastest known *randomized* algorithm for the problem, developed by Cygan et al., runs in $O(3^k n^{O(1)})$ time [10].

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 [11–13] — and Odd Cycle Transversal — Independent Odd Cycle Transversal [14, 15] — 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 [16], while its connected variant CFVS does not admit a kernel of any polynomial size (under certain complexity-theoretic assumptions) [17]. 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.

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 deterministic algorithm for FVS which runs in $O^\star(5^k)$ time [18]. 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 $CoNP \subseteq NP/Poly$ [17]. Our kernelization procedure makes use of the q-expansion lemma (See Lemma 1 on page 14), a generalization of Hall's Theorem, with q set to $k + 2$.

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 degree at most one and

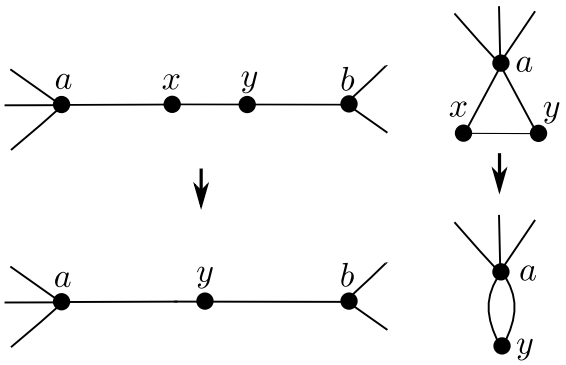

Figure 1: Bypassing a degree-two vertex which has another such vertex adjacent to it.

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 [16] — 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.

Claim 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*.*

Proof. Let F be a minimal IFVS of G of size at most k. If $F \cap \{x, y\} = \emptyset$, then F survives intact in G', and F itself is an IFVS of G' of size at most k. So let $F \cap \{x, y\} \neq \emptyset$. Observe that in G, the set of cycles which pass through x is exactly the same as the set of cycles which pass through y, and all these cycles pass through both a and b. If $F \cap \{a, b\} \neq \emptyset$, then $F' = F \setminus \{x, y\}$ is a strictly smaller IFVS of G, which contradicts the minimality of F. So $F \cap \{a, b\} = \emptyset$.

Observe that since $\{x, y\}$ is an edge in G, at most one of $\{x, y\}$ is in F. Thus we have $|F \cap \{x, y\}| = 1$ and $F \cap \{a, b\} = \emptyset$. If $x \in F$, then let $F' = (F \setminus \{x\}) \cup \{y\}$; otherwise, let $F' = F$. In either case, F' is an IFVS of G' of size at most k.

Conversely, let F' be an IFVS of G' of size at most k. Since we can obtain G from G' without (i) deleting vertices, or (ii) introducing a new edge between any two vertices of G', F' is an independent set in G as well, of size at most k. Every cycle which is present in G is also present in G' (though those which pass through x are shorter by one edge), and so F' is a feedback vertex set of G as well. Thus F' is an IFVS of G of size at most k. □

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 whether G has an FVS of size at most k, by invoking as a subroutine the algorithm due to Chen et al. [9]. 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 $N = F \setminus Y$: see Figure 2.

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 (See Figure 3). 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 of

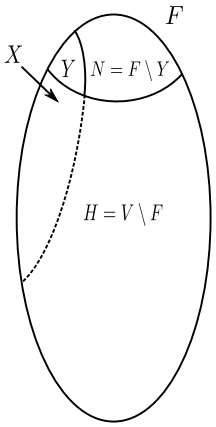

Figure 2: The search routine guesses the set $Y = X \cap F; 0 \leq |Y| \leq k$, where X is an unknown IFVS of the graph, of size at most k.

Algorithm 1. See figures 4 and 5.

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.

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

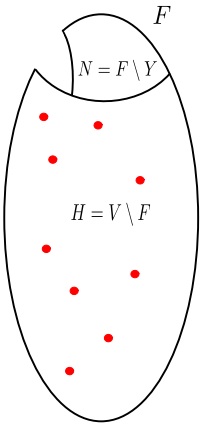

Figure 3: After guessing Y, the search routines colors all neighbors of Y in H red, and deletes the set Y. At this point, both the graphs $G[N]$ and $G[H]$ are forests.

Algorithm 1 BRANCH(G, H, N), Step 1. See Figure 4 and the 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.
```

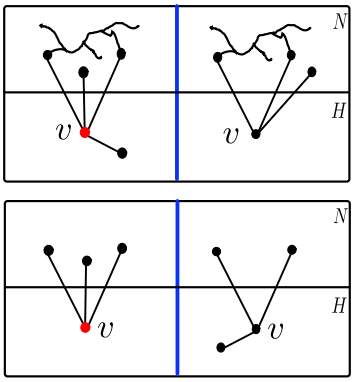

Figure 4: The cases handled by Algorithm 1, Step 1.

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.

Algorithm 1 BRANCH(G, H, N), Step 2. See Figure 5.

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.

"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.

Algorithm 1 BRANCH(G, H, N), Step 3. See Figure 5.

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.

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

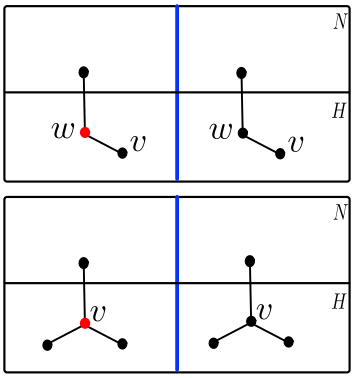

Figure 5: The cases handled by Algorithm 1, Steps 2 and 3.

$W \subseteq H$ of white vertices that forms an 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. [18, 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. *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.*

Proof. We prove the claim by first showing that every tree in $G'[H]$ is a single vertex. If a tree in $G'[H]$ consists of a single edge $\{u, v\}$, then since both u and v have degree at least two (Reduction Rule 1), both u and v have at least one neighbor in N. Neither u nor v can have *more than* one neighbor in N, or else Step 1 in the branching would have applied. So both u and v have exactly one neighbor in N, and so both have degree exactly two in the whole graph. This contradicts the fact that the graph is reduced with respect to Reduction Rule 2.

If a tree T in $G'[H]$ is not a single vertex or a single edge, then it has at least three vertices, and so contains a vertex of degree at least two. Now we show that such a vertex cannot exist. For, let v be a vertex of degree at least two in $G'[H]$, and let x, y be two of its neighbors in $G'[H]$. Both x and y cannot be leaves in $G'[H]$, or else Step 3 would have applied. So let x be a non-leaf in $G'[H]$. We now root the tree T at the vertex v. Let a be a leaf in the subtree rooted at x which is the farthest from v, and let b be its parent. Because of Reduction Rule 1 and Step 1, a has

degree exactly two. The vertex b cannot have another child in T — if this child is a leaf in $G'[H]$ then Step 3 would have applied, and otherwise a is not a farthest leaf in the subtree rooted at x. Thus b has exactly one other neighbor in $G'[H]$, namely its parent in the tree T. The vertex b has no neighbor in N, or else Step 2 would applied. Thus a and b are adjacent vertices with degree exactly two in the whole graph, contradicting the fact that the graph is reduced with respect to Reduction Rule 2.

It follows that every tree in $G'[H]$ is indeed a single vertex. Each such vertex has degree at least two (Reduction Rule 1), and all its neighbors are in N. It follows from Step 1 that all these vertices have exactly two neighbors in N, and the claim follows. □

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. *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'.*

Proof. Observe that the set Y' is initialized to the set Y and the budget b is initially set to $k - |Y|$. Further, b is decremented by exactly the number of vertices which are picked to be added to Y'. Note also that no step in the routine takes vertices out of the set Y'. It follows that a step which makes b negative raises the cardinality of Y' above k. Since no later step can reduce the size of Y', the claim with respect to b follows.

Now consider a point in the algorithm (call this point A) where $\mu = b + c - u$ becomes negative. Then u becomes greater than $b + c$, and since (i) no step in the algorithm decrements u, and (ii) the only steps that increment $b + c$ — namely, Line 19 and the branch at Line 21 which picks v to be not in the solution — also increment u by a larger amount, any later step maintains the inequality $u > b + c$. In particular, this inequality holds at every leaf of the recursion tree rooted at A. Let G' be the graph obtained at any one such leaf, and let N', H', b', c', u' be the values of N, H, b, c, u, respectively, in G'. By Claim 2, every vertex in H' has degree exactly two in G', and both its neighbors are in N'. Observe that the subgraph of G' induced by N' is a forest, and that no vertex in N' has both its neighbors in one tree in this forest. Consider the graph $\tilde{G}$ obtained

from G' by replacing (i) each component of $G'[N']$ by a single vertex, and (ii) each vertex v in H' by an edge between the vertices corresponding to the two components to which v has an edge. $\tilde{G}$ has c vertices and u edges, and so the smallest feedback *arc* set of $\tilde{G}$ has $u - c + 1 > b$ edges (since a tree on c vertices has $c - 1$ edges). From the construction, every feedback arc set of $\tilde{G}$ naturally corresponds to a FVS of G' of the same size chosen from H', and vice versa. It follows that any FVS of G' chosen from H' has more than b vertices; that is, more than what the budget allows. Since this argument holds for every leaf of the recursion tree rooted at A, the claim with respect to μ follows. □

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

Proof. Observe that $\mu \leq 2k$ at the start of the branching. Note that μ never increases, and decreases by at least one on each branch. Refer to Algorithm 1:

The non-branching steps. Line 6: μ decreases by one because b decreases by one and c, u do not change. Line 9: μ decreases by at least one because c decreases by at least one, b does not change, and u does not decrease (it may increase). Line 14: observe that v has degree at least two in the graph, or else it would have been deleted by Reduction Rule 1. Since v is white, it has no neighbor in Y. Since it is a leaf in $G'[H]$, it has at least one neighbor in N. Since v was not moved to Y or N by the previous branching rule, it has at most one neighbor in N. Thus v had exactly one neighbor in N before the current step, and so it becomes a useful vertex after this step. Thus μ decreases by one because u increases by one, and the other components of μ do not change. Line 19: b does not change, and c may increase by at most one. By similar reasoning as for Line 14, all the leaves adjacent to v in $G'[H]$ become useful vertices. Since there are at least two such leaves, μ decreases by at least one.

The branching steps. Line 11: If v is picked in the solution, then μ decreases by one because b decreases by one and c, u do not change. If v is not picked in the solution, μ decreases by at

least one because c decreases by at least one, b does not change, and u does not decrease (it may increase, causing μ to decrease further). Line 16: If w is picked in the solution, then μ decreases by one because b decreases by one and c, u do not change. If w is not picked in the solution, then μ decreases by one by similar reasoning as for Line 14. Line 21: If v is picked in the solution, then μ decreases by one because b decreases by one and c, u do not change. If v is not picked in the solution, then μ decreases by at least one by similar reasoning as for Line 19.

For a given choice of the set $Y; |Y| = i, 0 \leq i \leq k$, we have $b = k - i, c \leq k + 1 - i$, so that $\mu \leq b + c \leq 2k - 2i$. Each branching step in the algorithm (Lines 11, 16, and 21 in Algorithm 1) is a two-way branch, and on each branch μ drops by at least one. Since the recursion stops when $\mu = 0$, the number of vertices in the recursion tree is bounded by $2 \cdot 2^{2k-2i} = 4^{k-i}$. Since each step can be executed in polynomial time, the total time taken for this choice of Y is $O^\star(4^{k-i})$, where the $O^\star$ notation hides polynomial factors in the running time. The running time of the search routine, taken over all choices of the set Y is thus $\sum_{i=0}^{k} \binom{k}{i} O^\star(4^{k-i}) = O^\star(5^k)$. The FVS algorithm initially invoked as a subroutine contributes an *additive* factor of $O^\star(3.83^k)$ to the running time, and so the total running time of the algorithm is $O^\star(5^k)$. □

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.

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 3. 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 4. *Let (G, k) be an instance of* IFVS*, and let $(H, k-1)$ 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 $k-1$.*

Proof. Let v be a vertex which is at the center of a $k+1$-flower in the graph G. By definition, any FVS F of G contains at least one vertex from each of the $k+1$ cycles passing through v. Since v is the only vertex which lies on any two of these cycles, if $v \notin F$, then F contains a *distinct* vertex from each of the $k+1$ cycles, and so $|F| \geq k+1$. It follows that if $|F| \leq k$, then $v \in F$, and so the rule is safe when it returns NO.

If the graph G has a non-red IFVS I of size at most k, then by the above argument, $v \in I$, and so no neighbor of v is in I. It follows that $I \setminus \{v\}$ is a non-red IFVS of $H = G \setminus \{v\}$ of size at most $k-1$. Conversely, if the graph H has a non-red IFVS I of size at most $k-1$, then $I \cup \{v\}$ is (i) an FVS of G, (ii) an independent set in G since every neighbor of v in G is red in H, and (iii) has size at most k, and the claim follows. □

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

Reduction Rule 4.

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.

Theorem 2. [19, 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 [19]; we use a stricter version due to Fomin et al. [20]. 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) $|F_x| = q$; (b) for any vertex $y \in S; y \neq x$, $F_x \cap F_y = \emptyset$. 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 ([20])**. [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 4, 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. We make this observation more precise:

Claim 5. *Let (G, k) be an instance of* IFVS *where G is reduced with respect to Reduction Rules 1 to 4. 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.*

Proof. Since G is reduced with respect to Reduction Rules 1 to 4, the vertex v satisfies the conditions of Theorem 2. We find the set X_v of size at most $2k$ in polynomial time, as per the theorem. Let $C_1, C_2, \ldots, C_p$ be the connected components of $G \setminus (X_v \cup \{v\})$.

Observe that there is at most one edge from v to each of these components; if there are two or more edges from v to some C_i, then the subgraph induced by $\{v\} \cup C_i$ contains a cycle, a contradiction since X_v does not intersect such a cycle. If more than k of the components contain a cycle, then we can stop the reduction and return NO, since no FVS of size at most k can intersect all these cycles. Because of Reduction Rule 3 there are at most k vertices in X_v to which there are multiple edges from v, and because of Reduction Rule 4 the maximum multiplicity of such an edge is two. There is at most one edge from v to each remaining vertex in X_v, and so at most $4k$ of the edges incident on v are "used up" in these ways. Thus there are at least $(k + 2)2k$ components

such that (i) there is exactly one edge from v to each component, and (ii) each component is a tree. Without loss of generality, let this set of components be $C_1, C_2, \ldots, C_\ell$. In each of these components, there is at least one vertex which is adjacent to some vertex in X_v. For, if no vertex in C_i is adjacent to any vertex in X_v, then in G the component C_i forms a tree, and there is exactly one edge which has one end in $V(G) \setminus V(C_i)$ and the other in $V(C_i)$: namely, the edge from v to C_i. It follows that C_i contains a vertex of degree one in G, a contradiction since G is reduced with respect to Reduction Rule 1.

Consider the bipartite graph $J = (X_v \uplus \{C_1, C_2, \ldots, C_\ell\}, E)$ where $\{x, C_i\} \in E; x \in X_v$ if and only if there is an edge in G from x to some vertex in C_i. Observe that J satisfies the conditions of the q-Expansion Lemma (Lemma 1) with $q = k+2$. Thus we can find, in polynomial time, nonempty subsets $S \subseteq X_v, T \subseteq \{C_1, C_2, \ldots, C_\ell\}$ such that in J, each vertex in S has a distinct $(k+2)$-star, and the neighborhood of T is the set S. Let C be the set of components corresponding to the set T. The vertex v and the sets S, C satisfy the conditions of the claim. □

Given a vertex v and a set S as in Claim 5, it can be shown (See the proof of Claim 6) 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 5. Let v be a vertex in G, let $S \subseteq V(G) \setminus \{v\}$, and let C be a set of (not necessarily all) components of $G \setminus S \cup \{v\}$ which satisfy the conditions of Claim 5. Color the neighbors of v in the components in 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 [16]. The only difference here is that we need $k + 2$ "private" (in the sense stated in Claim 5) 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 6. *Let G, k, H be as in the description of Reduction Rule 5. 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.*

Proof. Let v, S, C be as in the description of the reduction rule. Let s be an arbitrary vertex in S, and let a, b be the red vertices associated with s as described in the rule.

We first show that if I is an inclusion-minimal non-red IFVS of G of size at most k, then I itself is a non-red IFVS of H. We consider two cases, $v \in I$ and $v \notin I$.

Note that $H \setminus \{v\}$ is identical to $G \setminus \{v\}$, except that (i) some more vertices are colored red in $H \setminus \{v\}$, and (ii) $H \setminus \{v\}$ may contain some extra red vertices of degree one attached to one or more vertices in S. Therefore, if $v \in I$, then since $G \setminus I$ contains no cycles, it follows that $H \setminus I$ contains no cycles either. Now observe that every vertex which is red in H and not in G is a neighbor of v in G, and every edge which is present in H and not in G has at least one of its end points as a neighbor of v in G. Since no neighbor of v in G is present in I, it follows that I is a non-red independent set in H as well.

If $v \notin I$, then $S \subseteq I$. To see this, observe that there are at least $k+2$ distinct paths from v to s in G, where the intermediate vertices in each path lie in a distinct component of C. If $v, s \notin I$, then since $|I| \leq k$, at least two of these paths survive in $G \setminus I$, and so form a cycle in $G \setminus I$, contradicting the fact that I is an FVS of G. Since s is an arbitrary vertex in S, it follows that $S \subseteq I$. Observe further that each edge from v to a component in C is a cut-edge in $G \setminus S$, and so does not belong to any cycle in $G \setminus S$. Since I is a minimal IFVS and $S \subseteq I$, I does not contain the other end-points of any of these edges. Thus I is a non-red independent set in H as well. Also, since every cycle which is present in H and not in G has at least one vertex in S, $H \setminus I$ contains no cycles.

Thus in both cases, I is a non-red IFVS of G of size at most k.

Now we show that if H has a non-red IFVS I of size at most k, then I itself is a non-red IFVS of G. We consider two cases, $v \in I$ and $v \notin I$.

If $v \in I$, then since $H \setminus \{v\}$ is nearly identical to $G \setminus \{v\}$ as noted above and $H \setminus I$ contains no cycles, it follows that $G \setminus I$ contains no cycles either. Now observe that every edge which is present in G and not in H has at least one of its end points red in H. Since no red vertex in H is present in I, it follows that I is an independent set in G as well.

If $v \notin I$, then $S \subseteq I$. For, if $v, s \notin I$, then these together with the red vertices a, b form a cycle of length four in $H \setminus I$, contradicting the fact that I is an FVS of H. Since s is an arbitrary vertex in S, it follows that $S \subseteq I$. G does not contain more vertices than H, and the only edges which are present in G and not in H are those from v to the components in C. Since $v \notin I$, no such edge is present in $G[I]$, and so I is an independent set in G. Further each such edge is a cut-edge in $G \setminus S$, and so does not belong to any cycle in $G \setminus S$. It follows that I is an FVS of G as well.

Thus in both cases, I is a non-red IFVS of G 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 7. *By repeatedly applying Reduction Rules 1 to 5 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.*

Proof. Let $n = |V(G)|, m = |E(G)|$. Observe first that each reduction rule either returns a YES/NO answer or decrements the sum $n + m$ by at least one. This is evident for Reduction Rules 1–4. Note that Reduction Rule 5 deletes at least $|S|(k + 2)$ edges and adds at most $2|S|$ vertices and at most $4|S|$ edges. Since we can assume without loss of generality that $k > 4$ (otherwise we can solve the problem in $O(n^4)$ time using brute-force), it follows that this rule also decrements $n + m$. So the *number* of applications of all the rules put together is at most $2(n + m)$.

It is not difficult to see that straightforward implementations of Reduction Rules 1, 2 and 4 run in polynomial time. Thomassé has shown [16] that Reduction Rules 3 and 5 can be applied in polynomial time. Thus every rule can be applied in polynomial time, and the claim follows. □

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

Claim 8. *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 5. 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.*

Proof. Let (G, k) be a YES instance of IFVS. Then $k' \leq k$, since no reduction rule increments k. From the correctness of the reduction rules, it follows that the graph H has a non-red IFVS I of size at most $k' \leq k$. Let $F = H[V(H) \setminus I]$ be the forest obtained by removing I from H. Since H is reduced with respect to Reduction Rule 5, the degree of a vertex in I (in fact, anywhere in H) is less than $4k + (k+2)2k$. Thus there are at most $4k^2 + (k+2)2k^2$ edges incident on the vertices in I, and so the number of vertices in F which have a neighbor in I is at most $4k^2 + (k+2)2k^2$. In particular, since H is reduced with respect to Reduction Rule 1, every leaf in F has at least one neighbor in I, and so there are at most $4k^2 + (k+2)2k^2$ leaves in F.

We now bound the number of vertices in F which do not have a neighbor in I. The number of non-leaf vertices in F of degree at least three is bounded by the number of leaves, and so is at most $4k^2 + (k+2)2k^2$. Let X be the set of non-leaf vertices in F of degree at most two which have no neighbor in I. Note that X is exactly the set of vertices of degree two in H which are internal nodes in F. We claim that $|X|$ is at most twice the number of leaves in F. To see this, root each tree in F at an arbitrary vertex. Map each vertex $x \in X$ to a nearest descendant in its tree which has degree at least three in the tree; if there is no such descendant, map x to a nearest leaf in the tree. Since H is reduced with respect to Reduction Rule 2 this map is injective, and the claim follows from the bound on the number of internal vertices of degree at least three in F.

Thus the total number of vertices in H is at most $k + 16k^2 + 8(k+2)k^2$. As seen above, the number of edges with one end point in I is at most $4k^2 + (k+2)2k^2$. Since F is a forest on at most $16k^2 + 8(k+2)k^2$ vertices, the number of edges in F is less than $16k^2 + 8(k+2)k^2$, and so the total number of edges in H is at most $20k^2 + 9(k+2)k^2$. This completes the proof. □

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

Reduction Rule 6. Let (G, k) be an instance of IFVS, and let (H, k') be the instance obtained by exhaustively applying Reduction Rules 1–5. 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').

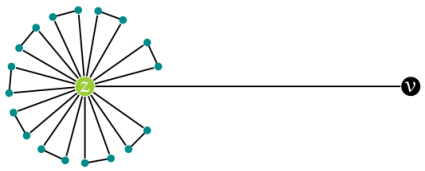

Figure 6: Removing the colors.

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 9. *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.*

Proof. Recall that a vertex was colored red to denote that it must not be picked in any solution. To remove this coloring, we add a gadget which forces each red vertex out of any solution, by making it adjacent to a vertex that must be in every solution. Formally, we add $2k' + 3$ new vertices $z, x_1, y_1, x_2, y_2, \ldots, x_{k'+1}, y_{k'+1}$ and the edges $\{x_i, y_i\}, \{x_i, z\}, \{y_i, z\}; 1 \leq i \leq k' + 1$. For each vertex v which is colored red in G', we add the edge $\{v, z\}$ and remove the color of v; see Figure 6. Let H be the resulting uncolored graph. Then $(H, k' + 1)$ is the final uncolored instance of IFVS. The correctness of this step follows from the fact that the new vertex z is at the center of a $k' + 1$-flower, and so it must be present in *any* FVS of G' of size at most k'. □

Putting all these together, we have:

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

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 [9] and has a quadratic kernel [16]. The connected variant, CFVS, is FPT but does not admit any polynomial kernel unless $CoNP \subseteq NP/Poly$ [17]. 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 [21]. 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.*

Proof. We reduce from the W[1]-hard Independent Set problem. Let (G, k) be an instance of Independent Set. We first describe the construction of (H, k), an instance of IDFVS based on (G, k) (See Figure 7).

Let $V(G) := \{v_1, \ldots, v_n\}$ denote the vertex set of G. The vertex set of H comprises of k "copies" of $V(G)$, that is, $V(H)$ has kn vertices. For ease of description, we use

$$u_i[j], 1 \leq i \leq n \text{ and } 1 \leq j \leq k$$

to denote the j^{th} copy of the vertex $v_i \in V(G)$, and $U[j]$ is used to refer to the set of vertices $\{u_1[j], \ldots, u_n[j]\}$.

The adjacencies among vertices of H are as follows:

- A directed cycle is induced on the vertices $U[j]$, for all $1 \leq j \leq k$. This involves the edges:

$$\{(u_1[j], u_2[j]), \cdots, (u_i[j], u_{i+1}[j]), \cdots, (u_n[j], u_1[j])\}.$$

- If (v_r, v_s) is an edge for $1 \leq r, s \leq n$ in G, then we add the edges:

$$(u_r[p], u_s[q]), (u_s[p], u_r[q])$$

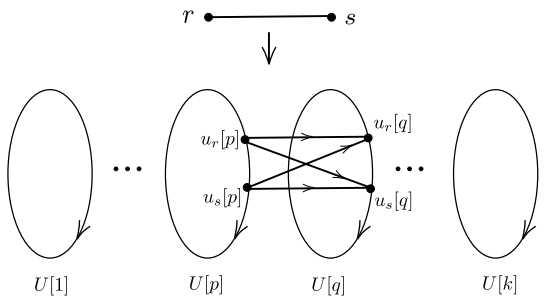

Figure 7: Reduction from Independent Set to Independent Directed Feedback Vertex Set

for all $1 \leq p < q \leq k$. Note that every edge of the original graph is represented twice between every pair of copies, and if $p < q$, then the edges between $U[p]$ and $U[q]$ point from $U[p]$ to $U[q]$.

- We also add edges of the form:

$$(u_i[p], u_i[q]), \quad \forall 1 \leq i \leq n, \text{ and } 1 \leq p < q \leq k.$$

This completes the construction of the reduced instance H. We make one remark that will have a bearing on the proof of correctness of the reduction. It is easily checked that the only directed cycles of H are the cycles induced on the vertex sets $U[j]$, for all $1 \leq j \leq k$. Indeed, notice that graph H induced on $U[j]$ is exactly a directed cycle, and if we contract every $U[j]$ to a single vertex $u[j]$, then the ordering

$$u[1], \ldots, u[j]$$

is a topological ordering of the graph. Therefore, the only cycles in H are exactly the k cycles induced on each $U[j]$.

We now turn to the proof. Let S be an independent subgraph of G of size at least k and let $\{v_{i_1}, \ldots, v_{i_k}\}$ be k elements from S (chosen arbitrarily in the event that S has more than k vertices). We claim that the vertices

$$S_H := \{u_{i_1}[1], \ldots, u_{i_k}[j]\}$$

form an IDFVS of H. Since $S_H \cap U[j] \neq \emptyset$ for all $1 \leq j \leq k$, S_H is a DFVS. It is independent because if there was an edge between, say, $u_{i_p}[p]$ and $u_{i_q}[q]$, then there would be an edge between v_{i_p} and v_{i_q} (this is by construction). This would contradict the fact that S induces an independent subgraph of G. Therefore, S_H is an IDFVS of H.

On the other hand, let S be an IDFVS of H. Since $|S| \leq k$, and H induces a directed cycle on $U[j]$ for all $1 \leq j \leq k$, we have that $S \cap U[j] \neq \emptyset$ for all $1 \leq j \leq k$. Let the vertex from $U[j]$ that belongs to S be $u_{i_j}[j]$. Note that S induces an independent subgraph, and recall that there is an edge between $u_i[p]$ and $u_i[q]$ for all $1 \leq i \leq n$, and $1 \leq r < s \leq k$. Therefore, we have that

$$i_p \neq i_q \forall 1 \leq p < q \leq k.$$

This implies that

$$v_{i_1} \neq v_{i_2} \neq \cdots \neq v_{i_k},$$

and that $S_H = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ consists of k distinct vertices. It is also clear that S_H induces an indpendent subgraph of G: if v_{i_p} and v_{i_q}, for example, are adjacent in G, then $u_{i_p}[p]$ and $u_{i_q}[q]$ are adjacent in S, which we assumed to be an independent subgraph of H, and this would be a contradiction. Therefore, S is an independent set of size k in G. □

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

[1] P. Festa, P. M. Pardalos, M. G. Resende, Feedback set problems, in: Handbook of Combinatorial Optimization, Kluwer Academic Publishers, 1999, pp. 209–258.

[2] R. G. Downey, M. R. Fellows, Parameterized Complexity, Springer-Verlag, New York, 1999.

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

[4] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*, Oxford University Press, Oxford, 2006.

[5] J. Guo, R. Niedermeier, Invitation to data reduction and problem kernelization, SIGACT News 38 (2007) 31–45.

[6] H. L. Bodlaender, Kernelization: New Upper and Lower Bound Techniques, in: J. Chen, F. V. Fomin (Eds.), Parameterized and Exact Computation: 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers, volume 5917 of *LNCS*, Springer, 2009, pp. 17–37.

[7] H. L. Bodlaender, On disjoint cycles, in: G. Schmidt, R. Berghammer (Eds.), 17th International Workshop on Graph–Theoretic Concepts in Computer Science, WG '91, Fischbachau, Germany, June 17-19, 1991, Proceedings, volume 570 of *LNCS*, Springer, 1992, pp. 230–238.

[8] R. G. Downey, M. R. Fellows, Fixed Parameter Tractability and Completeness, in: Complexity Theory: Current Research, Cambridge University Press, 1992, pp. 191–225.

[9] Y. Cao, J. Chen, Y. Liu, On Feedback Vertex Set: New Measure and New Structures, in: H. Kaplan (Ed.), Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010, volume 6139 of *LNCS*, Springer, 2010, pp. 93–104.

[10] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, J. O. Wojtaszczyk, Solving connectivity problems parameterized by treewidth in single exponential time, 2011. Accepted at the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011).

[11] F. V. Fomin, F. Grandoni, D. Kratsch, Solving connected dominating set faster than 2^n, Algorithmica 52 (2008) 153–166.

[12] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (1998) 374–387.

[13] D. Lokshtanov, M. Mnich, S. Saurabh, Linear Kernel for Planar Connected Dominating Set, in: J. Chen, S. B. Cooper (Eds.), Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009, Changsha, China, May 18-22, 2009, volume 5532 of *LNCS*, pp. 281–290.

[14] B. A. Reed, K. Smith, A. Vetta, Finding odd cycle transversals, Operations Research Letters 32 (2004) 299–301.

[15] D. Marx, B. O'Sullivan, I. Razgon, Treewidth reduction for constrained separation and bipartization problems, in: J.-Y. Marion, T. Schwentick (Eds.), 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France, volume 5 of *LIPIcs*, pp. 561–572.

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

[17] N. Misra, G. Philip, V. Raman, S. Saurabh, S. Sikdar, FPT algorithms for connected feedback vertex set, Journal of Combinatorial Optimization (2011, published online.) DOI:10.1007/s10878–011–9394–2.

[18] J. Chen, F. V. Fomin, Y. Liu, S. Lu, Y. Villanger, Improved Algorithms for Feedback Vertex Set Problems, Journal of Computer and System Sciences 74 (2008) 1188–1198.

[19] S. Thomassé, A quadratic kernel for feedback vertex set, in: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), Society for Industrial and Applied Mathematics, 2009, pp. 115–119.

[20] F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, S. Saurabh, Hitting forbidden minors: Approximation and kernelization, in: T. Schwentick, C. Dürr (Eds.), 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany, volume 9 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 189–200.

[21] J. Chen, Y. Liu, S. Lu, B. O'sullivan, I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem, Journal of the ACM 55 (2008) 21:1–21:19.