Exploring the Kernelization Borders for Hitting Cycles

Akanksha Agrawal
Institute of Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
akanksha@sztaki.mta.hu

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

Pranabendu Misra
University of Bergen, Bergen, Norway
Pranabendu.Misra@uib.no

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

Abstract

A generalization of classical cycle hitting problems, called conflict version of the problem, is defined as follows. An input is undirected graphs G and H on the same vertex set, and a positive integer k, and the objective is to decide whether there exists a vertex subset $X \subseteq V(G)$ such that it intersects all desired "cycles" (all cycles or all odd cycles or all even cycles) and X is an independent set in H. In this paper we study the conflict version of classical Feedback Vertex Set, and Odd Cycle Transversal problems, from the view point of kernelization complexity. In particular, we obtain the following results, when the conflict graph H belongs to the family of d-degenerate graphs.

1. CF-FVS admits a $\mathcal{O}(k^{\mathcal{O}(d)})$ kernel.
2. CF-OCT does not admit polynomial kernel (even when H is 1-degenerate), unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

For our kernelization algorithm we exploit ideas developed for designing polynomial kernels for the classical Feedback Vertex Set problem, as well as, devise new reduction rules that exploit degeneracy crucially. Our main conceptual contribution here is the notion of "k-independence preserver". Informally, it is a set of "important" vertices for a given subset $X \subseteq V(H)$, that is enough to capture the independent set property in H. We show that for d-degenerate graph independence preserver of size $k^{\mathcal{O}(d)}$ exists, and can be used in designing polynomial kernel.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Kernelization, Conflict-free problems, Feedback Vertex Set, Even Cycle Transversal, Odd Cycle Transversal

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.14

Funding This research has received funding from the European Research Council under ERC grant no. 306992 PARAPPROX, ERC grant no. 715744 PaPaALG and ERC grant no. 725978 SYSTEMATIC-GRAPH, and DST, India for SERB-NPDF fellowship [PDF/2016/003508].

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 14; pp. 14:1–14:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Reducing the input data, in polynomial time, without altering the answer is one of the popular ways in dealing with intractable problems in practice. While such polynomial time heuristics can not solve NP-hard problems exactly, they work well on input instances arising in real-life. It is a challenging task to assess the effectiveness of such heuristics theoretically. Parameterized complexity, via kernelization, provides a natural way to quantify the performance of such algorithms. In parameterized complexity each problem instance comes with a parameter k and the parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm, 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. The reduced instance is called a $p(k)$ kernel for the problem.

The quest for designing polynomial kernels for "hitting cycles" in undirected graphs has played significant role in advancing the field of polynomial time pre-processing – kernelization. Hitting all cycles, odd cycles and even cycles correspond to well studied problems of Feedback Vertex Set (FVS), Odd Cycle Transversal (OCT) and Even Cycle Transversal (ECT), respectively. Alternatively, FVS, OCT and ECT correspond to deleting vertices such that the resulting graph is a forest, a bipartite graph and an odd cactus graph, respectively. All these problems, FVS, OCT, and ECT, have been extensively studied in parameterized algorithms and kernelization. The earliest known FPT algorithms for FVS go back to the late 80's and the early 90's [4, 11] and used the seminal Graph Minor Theory of Robertson and Seymour. On the other hand the parameterized complexity of OCT was open for long time. Only, in 2003, Reed et al. [24] gave a $3^k n^{\mathcal{O}(1)}$ time algorithm for OCT. This is also the paper which introduced the *method of iterative compression* to the field of parameterized complexity. However, the existence of polynomial kernel, for FVS and OCT were open questions for long time. For FVS, Burrage et al. [7] resolved the question in the affirmative by designing a kernel of size $\mathcal{O}(k^{11})$. Later, Bodlaender [5] reduced the kernel size to $\mathcal{O}(k^3)$, and finally Thomassé [25] designed a kernel of size $\mathcal{O}(k^2)$. The kernel of Thomassé [25] is best possible under a well known complexity theory hypothesis. It is important to emphasize that [25] popularized the method of *expansion lemma*, one of the most prominent approach in designing polynomial kernels. While, the kernelization complexity of FVS was settled in 2006, it took another 6 years and a completely new methodology to design polynomial kernel for OCT. Kratsch and Wahlström [16] resolved the question of existence of polynomial kernel for OCT by designing a randomized kernel of size $\mathcal{O}(k^{4.5})$ using matroid theory.[1] As a counterpart to OCT, Misra et al. [20] studied ECT and designed an $\mathcal{O}(k^3)$ kernel.

Fruitful and productive research on FVS and OCT have led to the study of several variants and generalizations of FVS and OCT. Some of these admit polynomial kernels and for some one can show that none can exist, unless some unlikely collapse happens in complexity theory. In this paper we study the following generalization of FVS, and OCT, from the view-point of kernelization complexity.

Conflict Free Feedback Vertex Set (CF-FVS) **Parameter:** k
Input: An undirected graph G, a conflict graph H on vertex set $V(G)$ and a non-negative integer k.
Question: Does there exist $S \subseteq V(G)$, such that $|S| \leq k$, $G - S$ is a forest and $H[S]$ is edgeless?

[1] This foundational paper has been awarded the Nerode Prize for 2018.

One can similarly define Conflict Free Odd Cycle Transversal (CF-OCT).

Motivation. On the outset, a natural thought is "*why does one care*" about such an esoteric (or obscure) problem. We thought *exactly the same* in the beginning, till we realized the modeling power the problem provides and the rich set of questions one can ask. In the course of this paragraph we will try to explain this. First observe that, if one wants to model "independent" version of these problems (where the solution is suppose to be an independent set), then one takes conflict graph to be same as the input graph. An astute reader will figure out that the problem as stated above is W[1]-hard – a simple reduction from Multicolor Independent Set with each color class being modeled as cycle and the conflict graph being the input graph. Thus, a natural question is: *when does the problem become* FPT? To state the question formally, let $\mathcal{F}$ and $\mathcal{G}$ be two families of graphs. Then, $(\mathcal{G}, \mathcal{F})$-CF-FVS is same problem as CF-FVS, but the input graph G and the conflict graph H are restricted to belong to $\mathcal{G}$ and $\mathcal{H}$, respectively. It immediately brings several questions: (a) for which pairs of families the problem is FPT; (b) can we obtain some kind of dichotomy results; and (c) what could we say about the kernelization complexity of the problem. We believe that answering these questions for basic problems such as FVS, OCT, and Dominating Set will extend both the tractability as well as intractability tools in parameterized complexity and led to some fruitful and rewarding research. It is worth to note that initially we were inspired to define these problems by similar problems in computational geometry. See related results for more on this.

Our Results and Methods. A graph G is called *d-degenerate* if every subgraph of G has a vertex of degree at most d. For a fixed positive integer d, let $\mathcal{D}_d$ denote the set of graphs of *degeneracy* at most d. In this paper we study the $(\star, \mathcal{D}_d)$-CF-FVS ($\mathcal{D}_d$-CF-FVS) problem. The symbol $\star$ denotes that the input graph G is arbitrary. One can similarly define $\mathcal{D}_d$-CF-OCT. In fact, we study, CF-OCT for a very restricted family of conflict graphs, a family of disjoint union of paths of length at most three and at most two star graphs. We denote this family as $\mathcal{P}^{\star\star}_{\leq 3}$ and this variant of CF-OCT as $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Starting point of our research is the recent study of Jain et al. [14], who studied conflict-free graph modification problems in the realm of parameterized complexity. As a part of their study they gave FPT algorithms for $\mathcal{D}_d$-CF-FVS, $\mathcal{D}_d$-CF-OCT and $\mathcal{D}_d$-CF-ECT using the independence covering families [17]. Their results also imply similar FPT algorithm when the conflict graph belongs to nowhere dense graphs. In this paper we focus on the kernelization complexity of $\mathcal{D}_d$-CF-FVS, and $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT obtain the following results.

1. $\mathcal{D}_d$-CF-FVS admits a $\mathcal{O}(k^{\mathcal{O}(d)})$ kernel.
2. $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT does not admit polynomial kernel, unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Note that $\mathcal{D}_0$ denotes edgeless graphs and hence $\mathcal{D}_0$-CF-FVS, and $\mathcal{D}_0$-CF-OCT are essentially FVS, and OCT, respectively. Thus, any polynomial kernel for $\mathcal{D}_d$-CF-FVS, and $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT, must generalize the known kernels for these problems. We remark that the above result imply that CF-FVS admits polynomial kernels, when the conflict graph belong to several well studied graph families, such as planar graphs, graphs of bounded degree, graphs of bounded treewidth, graphs excluding some fixed graph as a minor, a topological minor and graphs of bounded expansion etc. (all these graphs classes have bounded degeneracy).

Strategy for CF-FVS. Our kernelization algorithm for CF-FVS consists of the following two steps. The first step of our kernelization algorithm is a structural decomposition of the input graph G. This does not depend on the conflict graph H. In this phase of the algorithm, given an instance (G, H, k) of CF-FVS we obtain an equivalent instance (G', H', k') of CF-FVS such that:

- The minimum degree of G' is at least 2.
- The number of vertices of degree at least 3 in G' is upper bounded by $\mathcal{O}(k^3)$. Let $V_{\geq 3}$ denote the set of vertices of degree at least 3 in G'.
- The number of maximal degree 2 paths in G' is upper bounded by $\mathcal{O}(k^3)$. That is, $G' - V_{\geq 3}$ consists of $\mathcal{O}(k^3)$ connected components where each component is a path.

We obtain this structural decomposition using reduction rules inspired by the quadratic kernel for FVS [25]. As stated earlier, this step can be performed for any graph H. Thus the problem reduces to designing reduction rules that bound the number of vertices of degree 2 in the reduced graph. Note that we can not do this for any arbitrary graph H as the problem is W[1]-hard. Once the decomposition is obtained we can not use the known *reduction rules* for FVS. This is for a simple reason that in G' the only vertices that are not bounded have degree exactly 2 in G'. On the other hand for FVS we can do simple "short-circuit" of degree 2 vertices (remove the vertex and add an edge between its two neighbors) and assume that there is no vertices of degree two in the graph. So our actual contributions start here.

The second step of our kernelization algorithm bounds the degree two vertices in the graph G'. Here we must use the properties of the graph H. We propose new reduction rules for bounding degree two vertices, when H belongs to the family of d-degenerate graphs. Towards this we use the notion of d-degeneracy sequence, which is an ordering of the vertices in H such that any vertex can have at most d forward neighbors. This is used in designing a marking scheme for the degree two vertices. Broadly speaking our marking scheme associates a set with every vertex v. Here, set consists of " paths and cycles of G' on which the forward neighbors of v are". Two vertices are called similar if their associated sets are same. We show that if some vertex is not marked then we can safely contract this vertex to one of its neighbors. We then upper bound the degree two vertices by $\mathcal{O}(k^{\mathcal{O}(d)} d^{\mathcal{O}(d)})$, and thus obtain a kernel of this size for $\mathcal{D}_d$-CF-FVS.

At the heart of our kernelization algorithm is a combinatorial tool of "k-independence preserver". Informally, it is a set of "important" vertices for a given subset $X \subseteq V(H)$, that is enough to capture the independent set property in H. We show that for d-degenerate graph independence preserver of size $k^{\mathcal{O}(d)}$ exists, and can be used in designing polynomial kernel. This is our main conceptual contribution.

Strategy for CF-OCT. The kernelization lower bound is obtained by the method of cross-composition [6]. We first define a conflict version of the s-t-Cut problem, where H belongs to $\mathcal{P}^{\star\star}_{\leq 3}$. Then, we show that the problem is NP-hard and cross composes to itself. Finally, we give a parameter preserving reduction from the problem to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT, and obtain the desired kernel lower bound.

Related Work. In the past, the conflict free versions of some classical problems have been studied, e.g. for Shortest Path [15], Maximum Flow [21, 22], Knapsack [23], Bin Packing [12], Scheduling [13], Maximum Matching and Minimum Weight Spanning Tree [10, 9]. It is interesting to note that some of these problems are NP-hard even when their non-conflicting version is polynomial time solvable. The study of conflict free problems has also been recently initiated in computational geometry motivated by various applications (see [1, 2, 3]).

2 Preliminaries

Throughout the paper, we follow the following notions. Let G be a graph, $V(G)$ and $E(G)$ denote the vertex set and the edge set of graph G, respectively. Let n and m denote the number of vertices and the number of edges of G, respectively. Let G be a graph and $X \subseteq V(G)$, then $G[X]$ is the graph induced on X and $G - X$ is graph G induced on $V(G) \setminus X$. Let Δ denotes the maximum degree of graph G. We use $N_G(v)$ to denote the neighborhood of v in G and $N_G[v]$ to denote $N_G(v) \cup \{v\}$. Let E' be subset of edges of graph G, by $G[E']$ we mean the graph with the vertex set $V(G)$ and the edge set E'. Let $X \subseteq E(G)$, then $G - X$ is a graph with the vertex set $V(G)$ and the edge set $E(G) \setminus X$. Let Y be a set of edges on vertex set $V(G)$, then $G \cup Y$ is graph with the vertex set $V(G)$ and the edge set $E(G) \cup Y$. Degree of a vertex v in graph G is denoted by $deg_G(v)$. For an integer ℓ, we denote the set $\{1, 2, \ldots, \ell\}$ by $[\ell]$. A *path* $P = \{v_1, \ldots, v_n\}$ is an ordered collection of vertices such that there is an edge between every consecutive vertices in P and v_1, v_n are *endpoints* of P. For a path P by $V(P)$ we denote set of vertices in P and by $E(P)$ we denote set of edges in P. A *cycle* $C = \{v_1, \ldots, v_n\}$ is a path with an edge $v_1 v_n$. We define a *maximal degree two induced path in* G as an induced path of maximal length such that all vertices in path are of *degree exactly two in* G. An *isolated cycle* in graph G is defined as an induced cycle whose all the vertices are of degree exactly two in G. Let G' and G be graphs, $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$, then we say that G' is a *subgraph* of G. The subscript in the notations will be omitted if it is clear from the context.

A graph G has *degeneracy* d if every subgraph of G has a vertex of degree at most d. An ordering of vertices $\sigma : V(G) \to \{1, \cdots, n\}$ is is called a *d-degeneracy sequence* of graph G, if every vertex v has at most d neighbors u with $\sigma(u) > \sigma(v)$. A graph G is d-degenerate if and only if it has a d-degeneracy sequence. For a vertex v in d-degenerate graph G, the neighbors of v which comes *after (before)* v in d-degeneracy sequence are called *forward (backward) neighbors* of v in the graph G. Given a d-degenerate graph, we can find d-degeneracy sequence in linear time [18].

3 A Tool for Our Kernelization Algorithm

In this section, we give a tool, which we believe might be useful in obtaining kernelization algorithm for "conflict free" versions of various parameterized problems (admitting kernels), when the conflict graph belongs to the family of d-degenerate graphs. We particularly use this tool to obtain kernel for $\mathcal{D}_d$-CF-FVS (Section 4). For a parameterized problem Π, consider an instance (G, H, k) of its conflict free variant, Conflict Free Π. Then in the kernelization step where we want to bound the number of vertices, it is seemingly useful to be able to obtain a set of "important" vertices for a given subset $X \subseteq V(H)$ that will be enough to capture the independent set property in H. The above intuition becomes clear when we describe the kernelization algorithm for $\mathcal{D}_d$-CF-FVS.

To formalize the notion of "important" set of vertices, we give the following definition.

▶ **Definition 1.** For a d-degenerate graph H and a set $X \subseteq V(H)$, a *k-independence preserver* for (H, X) is a set $X' \subseteq X$, such that for any independent set S in H of size at most k, if there is $v \in (S \cap X) \setminus X'$, then there is $v' \in X' \setminus S$, such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in H.

Throughout this section, we work with a (fixed) d, which is the degeneracy of the input graph. The goal of this section will be to obtain an algorithm for computing a k-independence preserver for (H, X) of "small" size. To quantify the "small" size, we need the following definition.

▶ **Definition 2.** For each $q \in [d]$, we define an integer n_q as follows.

1. If $q = 1$, then $n_q = kd + k + 1$, and
2. $n_q = kn_{q-1} + kd + k + 1$, otherwise.

Next, we formally define the problem for which we want to design a polynomial time algorithm. We call this problem d-Bounded Independence Preserver (d-BIP, for short).

> d-Bounded Independence Preserver (d-BIP)
> **Input:** A d-degenerate graph H, a set $X \subseteq V(H)$, and an integer k.
> **Output:** A set $X' \subseteq X$ of size at most n_{d+1}, such that X' is a k independence preserver for (H, X).

In the following, let (H, X, k) be an instance of d-BIP. We work with a (fixed) d-degeneracy sequence, σ of H. We recall that such a sequence can be computed in polynomial time [18]. Forward and backward neighbors of a vertex v are also defined with respect to the ordering σ. If $\sigma(u) < \sigma(v)$, then u is a backward neighbor of v and v is a forward neighbor of u. By $N^f_H(v)$ ($N^b_H(v)$) we denote the set of forward (backward) neighbors of the vertex v in H.

To design our polynomial time algorithm for d-BIP, we need the notion of q-*reducible sets*, which is formally defined below.

▶ **Definition 3.** A set $Y \subseteq V(H)$ is q-*reducible*, if for every set $U \subseteq Y$, for which there is a set $Z \subseteq V(H)$, such that: (i) Z is of size exactly $d - q + 1$ and (ii) for each $u \in U$, we have $Z \subseteq N^f_H(u)$, it holds that $|U| \le n_q$.

Now, we give our polynomial time algorithm for d-BIP in Algorithm 1.

Algorithm 1 Algo1(H, X).

Require: d-degenerate graph H, $X \subseteq V(H)$, and an integer k.
Ensure: $X' \subseteq X$ of size at most n_{d+1}, which is a k-independence preserver of (H, X).

```
For q ∈ [d], set n_q = kd + 1, when q = 1, and n_q = kn_{q−1} + kd + k + 1, otherwise.
q = 1.
while q ≤ d do
    while X is not q-reducible do
        Find U ⊆ X of size n_q + 1, for which there is Z ⊆ V(H) of size exactly d − q + 1,
        such that for each u ∈ U, we have Z ⊆ N^f_H(u).
        Let v be an arbitrary vertex in U.
        X = X \ {v}.
    end while
    q = q + 1.
end while
while |X| > n_{d+1} do
    Let v be an arbitrary vertex in X.
    X = X \ {v}.
end while
Set X' = X.
return X'
```

To prove the correctness of our algorithm, we state an observation, the proof of which follows from the fact that any vertex can have at most d forward neighbors in H.

▶ **Observation 4.** *Let H be a d-degenerate graph and S be an independent set of H of size at most k. Then, for any set $U \subseteq V(H)$, such that for each vertex $u \in U$, $N^b_H(u) \cap S \neq \emptyset$, we have that $|U| \leq kd$.*

Now we are ready to prove the correctness of our algorithm (Algorithm 1) for d-BIP.

▶ **Lemma 5.** *Algorithm 1 is correct.*

Proof. Let (H, X, k) be an instance of d-BIP, and X' be the output returned by Algorithm 1 with it as the input. Clearly, $X' \subseteq X$ as we do not add any new vertex to obtain the set X', and size of X' is bounded by n_{d+1}, since at Step 10-13 of the algorithm we reduce its size to (at most) n_{d+1}. Therefore, it remains to show that X' is a k-independence preserver of (H, X). To this end, we consider the following cases.

Case 1: X is q-reducible, for each $q \in [d]$. In this case, the algorithm arbitrarily deletes vertices (if required) from X to obtain X'. If $X = X'$, then the claim trivially holds. Therefore, we assume that X' is a strict subset of X. To show that X' is a k-independence preserver for (H, X), consider an independent set S in H of size at most k. Furthermore, consider a vertex $v \in (S \cap X) \setminus X'$ (again, if such a vertex does not exists, the claim follows). To prove the desired result, we want to find a replacement vertex for v in X' which can be added to S (after removing v) to obtain an independent set in H. To this end, we mark some vertices in X'. Firstly, mark all the forward neighbors of each $s \in S$ in the set X'. That is, we let X'_M to be the set $(\cup_{s \in S} N^f_H(s)) \cap X'$. Also, we add all vertices in $S \cap X'$ to the set X'_M. By the property of d-degeneracy sequence, we have that $|X'_M| \leq kd + k$ (see Observation 4). Next, we will mark some more vertices in X'_M with the hope to find a replacement vertex for v in $X' \setminus X'_M$ to add to S. Recall that by our assumption X is q-reducible, for each $q \in [d]$, and in particular, it is d-reducible. Thus, for each $s \in S$, the set $X_s = \{x \in X \mid s \in N^f_H(x)\} \subseteq X$ has size at most n_d. Based on the above observation, we describe our second level of marking of vertices in X'. For each $s \in S$, we add each vertex in X_s to X'_M. From the discussions above, we have that $|X'_M| \leq kd + k + kn_d$. Since $|X'| = n_{d+1}$, and by definition, $n_{d+1} = kn_d + kd + k + 1$, we have $X' \setminus X'_M \neq \emptyset$. Moreover, no vertex in X' has a neighbor in $S \setminus \{v\}$. Therefore, for $v' \in X' \setminus X'_M$, we have that $S' = (S \setminus \{v\}) \cup \{v'\}$ is an independent set in H.

Case 2: X is not q-reducible, for some $q \in [d]$. Let q' be the smallest integer for which X is not q'-reducible. Since X is not q'-reducible, there is a set $U \subseteq X$ of size at least $n_q + 1$, for which there is a set $Z \subseteq V(H)$ of size exactly $d - q + 1$, such that for each $u \in U$, we have $Z \subseteq N^f_H(u)$. Consider (first) such pair of sets U, Z considered by the algorithm in Step 4. Furthermore, let $v \in U$ be the vertex deleted by the algorithm in Step 6. Let $\hat{U} = U \setminus \{v\}$. To prove the claim, it is enough to show that for an independent set S of size at most k containing v in H, there is $v' \in \hat{U}$ such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in H. Here, we will use the fact that deleting a vertex from a set does not change a set from being $\tilde{q}$-reducible to a set which is not $\tilde{q}$-reducible, where $\tilde{q} \in [d]$. In the following, consider an independent set S of size at most k containing v in H. We construct a marked set $\hat{U}_M$, of vertices in $\hat{U}$. Firstly, we add all the vertices in $(\cup_{s \in S \setminus \{v\}} N^f_H(s)) \cap \hat{U}$ to $\hat{U}_M$. Also, we add all vertices in $S \cap \hat{U}$ to $\hat{U}_M$. Notice that at the end of above marking scheme, we have $|\hat{X}_M| \leq kd + k$. We will mark some more vertices in $\hat{U}$. Before stating the second level of marking, we remark that $S \cap Z = \emptyset$. For each $s \in S \setminus \{v\}$, let $Z_s = Z \cup \{s\}$. Since $S \cap Z = \emptyset$, we have that $|Z_s| = d - (q - 1) + 1$. For $s \in S \setminus \{v\}$, let $\hat{U}_s = \{u \in \hat{U} \mid Z_s \subseteq N^f_H(u)\}$. Since X is q^*-reducible for each $q^* < q'$, we have $|\hat{U}_s| \leq n_{q-1}$, for each $s \in S \setminus \{v\}$. Now we are ready to describe our second level of marking. For each $s \in S \setminus \{v\}$, add all vertices in U_s to the set $\hat{U}_M$. Notice that $|\hat{U}_M| \leq kd + k + kn_{q-1}$. Moreover, $|\hat{U}| \geq n_q$ and $n_q = kn_{q-1} + kd + k + 1$. Thus, there is a vertex $v' \in \hat{U} \setminus \hat{U}_M$, such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in H. ◀

▶ **Lemma 6.** ($\star$)[2] *Algorithm 1 runs in time $n^{\mathcal{O}(d)}$.*

Using Lemma 5 and Lemma 6 we obtain the following theorem.

▶ **Theorem 7.** d-Bounded Independence Preserver *admits an algorithm running in time $n^{\mathcal{O}(d)}$.*

4 A Polynomial Kernel for $\mathcal{D}_d$-CF-FVS

In this section, we design a kernelization algorithm for $\mathcal{D}_d$-CF-FVS.

To design a kernelization algorithm for $\mathcal{D}_d$-CF-FVS, we define another problem called $\mathcal{D}_d$-Disjoint-CF-FVS ($\mathcal{D}_d$-DCF-FVS, for short). We first define the problem $\mathcal{D}_d$-DCF-FVS formally, and then explain its uses in our kernelization algorithm.

$\mathcal{D}_d$-Disjoint-CF-FVS ($\mathcal{D}_d$-DCF-FVS) **Parameter:** k
Input: An undirected graph G, a graph $H \in \mathcal{D}_d$ such that $V(G) = V(H)$, a subset $R \subseteq V(G)$, and a non-negative integer k.
Question: Is there a set $S \subseteq V(G) \setminus R$ of size at most k, such that $G - S$ does not have any cycle and S is an independent set in H?

Notice that $\mathcal{D}_d$-CF-FVS is a special case of $\mathcal{D}_d$-DCF-FVS, where $R = \emptyset$. Given an instance of $\mathcal{D}_d$-CF-FVS, the kernelization algorithm creates an instance of $\mathcal{D}_d$-DCF-FVS by setting $R = \emptyset$. Then it applies a kernelization algorithm for $\mathcal{D}_d$-DCF-FVS. Finally, the algorithm takes the instance returned by the kernelization algorithm for $\mathcal{D}_d$-DCF-FVS and generates an instance of $\mathcal{D}_d$-CF-FVS. Before moving forward, we note that the purpose of having set R is to be able to prohibit certain vertices to belong to a solution. This is particularly useful in maintaining the independent set property of the solution, when applying reduction rules which remove vertices from the graph (with an intention of it being in a solution).

We first focus on designing a kernelization algorithm for $\mathcal{D}_d$-DCF-FVS, and then give a polynomial time linear parameter preserving reduction from $\mathcal{D}_d$-DCF-FVS to $\mathcal{D}_d$-CF-FVS. If the kernelization algorithm for $\mathcal{D}_d$-DCF-FVS returns that (G, H, R, k) is a YES (NO) instance of $\mathcal{D}_d$-DCF-FVS, then conclude that (G, H, k) is a YES (NO) instance of $\mathcal{D}_d$-CF-FVS. In the following, we describe a kernelization algorithm for $\mathcal{D}_d$-DCF-FVS. Let (G, H, R, k) be an instance of $\mathcal{D}_d$-DCF-FVS. The algorithm starts by applying the following simple reduction rules.

▶ **Reduction Rule 1.**
(a) *If $k \geq 0$ and G is acyclic, then return that (G, H, R, k) is a* YES *instance of* $\mathcal{D}_d$-DCF-FVS.
(b) *Return that (G, H, R, k) is a* NO *instance of* $\mathcal{D}_d$-DCF-FVS*, if one of the following conditions is satisfied:*
  (i) *$k \leq 0$ and G is not acyclic,*
  (ii) *G is not acyclic and $V(G) \subseteq R$, or*

[2] Due to space constraints, the proofs of results marked with $\star$ have been omitted from this extended abstract.

(iii) *There are more than k isolated cycles in G.*

▶ **Reduction Rule 2.**

(a) *Let v be a vertex of degree at most 1 in G. Then delete v from the graphs G, H and the set R.*

(b) *If there is an edge in G (H) with multiplicity more than 2 (more than 1), then reduce its multiplicity to 2 (1).*

(c) *If there is a vertex v with self loop in G. If $v \notin R$, delete v from the graphs G and H, and decrease k by one. Furthermore, add all the vertices in $N_H(v)$ to the set R, otherwise return that (G, H, R, k) is a* NO *instance of* $\mathcal{D}_d$-DCF-FVS.

(d) *If there are parallel edges between (distinct) vertices $u, v \in V(G)$ in G:*

  (i) *If $u, v \in R$, then return that (G, H, R, k) is a* NO *instance of* $\mathcal{D}_d$-DCF-FVS.

  (ii) *If $u \in R$ ($v \in R$), delete v (u) from the graphs G and H, and decrease k by one. Furthermore, add all the vertices in $N_H(v)$ ($N_H(u)$) to the set R.*

It is easy to see that the above reduction rules are correct, and can be applied in polynomial time. In the following, we define some notion and state some known results, which will be helpful in designing our next reduction rules.

▶ **Definition 8.** For a graph G, a vertex $v \in V(G)$, and an integer $t \in \mathbb{N}$, a t-*flower* at v is a set of t vertex disjoint cycles whose pairwise intersection is exactly $\{v\}$.

▶ **Proposition 9.** [8, 19, 25] *For a graph G, a vertex $v \in V(G)$ without a self-loop in G, and an integer k, the following conditions hold.*

(i) *There is a polynomial time algorithm, which either outputs a $(k+1)$-flower at v, or it correctly concludes that no such $(k+1)$-flower exists. Moreover, if there is no $(k+1)$-flower at v, it outputs a set $X_v \subseteq V(G) \setminus \{v\}$ of size at most $2k$, such that X_v intersects every cycle passing through v in G.*

(ii) *If there is no $(k+1)$-flower at v in G and the degree of v is at least $4k + (k+2)2k$. Then using a polynomial time algorithm we can obtain a set $X_v \subseteq V(G) \setminus \{v\}$ and a set $\mathcal{C}_v$ of components of $G[V(G) \setminus (X_v \cup \{v\})]$, such that each component in $\mathcal{C}_v$ is a tree, v has exactly one neighbor in $C \in \mathcal{C}_v$, and there exist at least $k+2$ components in $\mathcal{C}_v$ corresponding to each vertex $x \in X_v$ such that these components are pairwise disjoint and vertices in X_v have an edge to each of their associated components.*

▶ **Reduction Rule 3.** *Consider $v \in V(G)$, such that there is a $(k+1)$-flower at v in G. If $v \in R$, then return that (G, H, R, k) is a* NO *instance of* $\mathcal{D}_d$-DCF-FVS. *Otherwise, delete v from G, H and decrease k by one. Furthermore, add all the vertices in $N_H(v)$ to R.*

The correctness of the above reduction rule follows from the fact that such a vertex must be part of every solution of size at most k. Moreover, the applicability of it in polynomial time follows from Proposition 9 (item (i)).

▶ **Reduction Rule 4.** *Let $v \in V(G)$, $X_v \subseteq V(G) \setminus \{v\}$, and $\mathcal{C}_v$ be the set of components which satisfy the conditions in Proposition 9(ii) (in G), then delete edges between v and the components of the set $\mathcal{C}_v$, and add parallel edges between v and every vertex $x \in X_v$ in G.*

The polynomial time applicability of Reduction Rule 4 follows from Proposition 9. And, in the following lemma, we prove the safeness of this reduction rule.

▶ **Lemma 10.** ($\star$) *Reduction Rule 4 is safe.*

In the following, we state an easy observation, which follows from non-applicability of Reduction Rule 1 to 4.

▶ **Observation 11.** *Let (G, H, R, k) be an instance of $\mathcal{D}_d$-DCF-FVS, where none of Reduction Rule 1 to 4 apply. Then the degree of each vertex in G is bounded by $\mathcal{O}(k^2)$.*

Proof. As Reduction Rule 3 is not applicable, then there is no $k+1$-flower in G. Now, if there is $v \in V(G)$ with degree at least $4k + (k+2)2k$, then Reduction Rule 4 would be applicable. ◀

To design our next reduction rule, we construct an auxiliary graph $G^\star$. Intuitively speaking, $G^\star$ is obtained from G by shortcutting all degree two vertices. That is, vertex set of $G^\star$ comprises of all the vertices of degree at least three in 3. From now on, vertices of degree at least 3 (in G) will be referred to as high degree vertices. For each $uv \in E(G)$, where u, v are high degree vertices, we add the edge uv in $G^\star$. Furthermore, for an induced maximal path P_{uv}, between u and v where all the internal vertices of P_{uv} are degree two vertices in G, we add the (multi) edge uv to $E(G^\star)$. Next, we will use the following result to bound the number of vertices and edges in $G^\star$.

▶ **Proposition 12.** [8] *A graph G with minimum degree at least 3, maximum degree Δ, and a feedback vertex set of size at most k has at most $(\Delta+1)k$ vertices and $2\Delta k$ edges.*

The above result (together with the construction of $G^\star$) gives us the following (safe) reduction rule.

▶ **Reduction Rule 5.** *If $|V(G^\star)| \geq 4k^2 + 2k^2(k+2)$ or $|E(G^\star)| \geq 8k^2 + 4k^2(k+2)$, then return* NO*.*

▶ **Lemma 13.** *Let (G, H, R, k) be an instance of $\mathcal{D}_d$-DCF-FVS, where none of the Reduction Rules 1 to 5 are applicable. Then we obtain the following bounds:*

- *The number of vertices of degree at least 3 in G is bounded by $\mathcal{O}(k^3)$.*
- *The number of maximal degree two induced paths in G is bounded by $\mathcal{O}(k^3)$.*

Having shown the above bounds, it remains to bound the number of degree two vertices in G. We start by applying the following simple reduction rule to eliminate vertices of degree two in G, which are also in R.

▶ **Reduction Rule 6.** *Let $v \in R$, $d_G(v) = 2$, and x, y be the neighbors of v in G. Delete v from the graphs G, H and the set R. Furthermore, add the edge xy in G.*

The correctness of this reduction rule follows from the fact that vertices in R can not be part of any solution and all the cycles passing through v also passes through its neighbors.

In the polynomial kernel for the Feedback Vertex Set problem (with no conflict constraints), we can short-circuit degree two vertices. But in our case, we cannot perform this operation, since we also need the solution to be an independent set in the conflict graph. Thus to reduce the number of degree two vertices in G, we exploit the properties of a d-degenerate graph. To this end, we use the tool that we developed in Section 3. This immediately gives us the following reduction rule.

▶ **Reduction Rule 7.** *Let P be a maximal degree two induced path in G. If $|V(P)| \geq n_{d+1}+1$, apply Algorithm 1 with input $(H, V(P) \setminus R)$. Let $\widehat{V}(P)$ be the set returned by Algorithm 1. Let $v \in (V(P) \setminus R) \setminus \widehat{V}(P)$, and x, y be the neighbors of v in G. Delete v from the graphs G, H. Furthermore, add edge xy in G.*

▶ **Lemma 14.** *Reduction Rule 7 is safe.*

Proof. Let (G, H, R, k) be an instance of $\mathcal{D}_d$-DCF-FVS and v be a vertex in a maximal degree two path P with neighbors x and y, with respect to which Reduction Rule 14 is applied. Furthermore, let (G', H', R, k) be the resulting instance after application of the reduction rule. We will show that (G, H, R, k) is a YES instance of $\mathcal{D}_d$-DCF-FVS if and only if (G', H', R, k) is a YES instance of $\mathcal{D}_d$-DCF-FVS.

In the forward direction, let (G, H, R, k) be a YES instance of $\mathcal{D}_d$-DCF-FVS and S be one of its minimal solution. Consider the case when $v \notin S$. In this case, we claim that S is also a solution of $\mathcal{D}_d$-DCF-FVS for (G', H', R, k). Suppose not then either S is not an independent set in H' or $G' - S$ contains a cycle. Since, H' is an induced subgraph of H, we have that S' is also an independent set in H'. So we assume that $G' - S$ has a cycle, say C. If C does not contain the edge xy, then C is also a cycle in $G - S$. Therefore, we assume that C contains the edge xy. But then $(C \setminus \{xy\}) \cup \{xv, vy\}$ is a cycle in $G - S$. Next, we consider the case when $v \in S$. By Lemma 5 we have a vertex $v' \in V(P) \setminus \{v\}$ such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in H'. By using the fact that any cycle that passes through v also contains all vertices in P (together with the discussions above) imply that $(S \setminus \{v\}) \cup \{v'\}$ is a solution of $\mathcal{D}_d$-DCF-FVS for (G', H', R, k).

In the reverse direction, let (G', H', R, k) be a YES instance of $\mathcal{D}_d$-DCF-FVS and S' be one of its minimal solution. We claim that S' is also a solution of $\mathcal{D}_d$-DCF-FVS for (G, H, R, k). Suppose not, then either S is not an independent set in H or $G - S$ contains a cycle. Since, H' is an induced subgraph of H, we have that S' is also an independent set in H. Next, assume that there is a cycle C in $G - S$. The cycle C must contain v, otherwise, C is also a cycle in $G' - S'$. Since v is a degree two vertex in G, therefore any cycle that contains v, must also contain x and y. As observed before, $G - \{xv, vy\}$ is identical to $G' - \{xy\}$. But then, $(C \setminus \{xv, vy\}) \cup \{xy\}$ is a cycle in $G' - S'$, a contradiction. This concludes that S' is a solution of $\mathcal{D}_d$-DCF-FVS for (G, H, R, k). ◀

▶ **Lemma 15.** ($\star$) *Let (G, H, R, k) be an instance of $\mathcal{D}_d$-DCF-FVS, where none of the Reduction Rules 1 to 7 are applicable. Then the number of vertices in a degree two induced path in G is bounded by $\mathcal{O}(k^{\mathcal{O}(d)})$.*

▶ **Theorem 16.** $\mathcal{D}_d$-DCF-FVS *admits a kernel with $\mathcal{O}(k^{\mathcal{O}(d)})$ vertices.*

▶ **Lemma 17.** ($\star$) *There is a polynomial time parameter preserving reduction from $\mathcal{D}_d$-DCF-FVS to $\mathcal{D}_d$-CF-FVS.*

By Theorem 16 and Lemma 17, we obtain the following result.

▶ **Theorem 18.** $\mathcal{D}_d$-CF-FVS *admits a kernel with $\mathcal{O}(k^{\mathcal{O}(d)})$ vertices.*

5 Kernelization Complexity of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT

In this section, we show that CF-OCT does not admit a polynomial kernel when the conflict graph belongs to the family $\mathcal{P}^{\star\star}_{\leq 3}$. Let $\mathcal{P}_{\leq 3}$ denotes the family of disjoint union of paths of length at most three, and $\mathcal{P}^{\star}_{\leq 3}$ denotes the family of disjoint union of paths of length at most three and a star graph. We give parameter preserving reduction from $\mathcal{P}^{\star}_{\leq 3}$-Conflict Free s-t Cut ($\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut) to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.

We first prove that $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut is NP-hard. Then, we prove that $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut does not admit a polynomial compression, unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$ using the method of cross-composition.

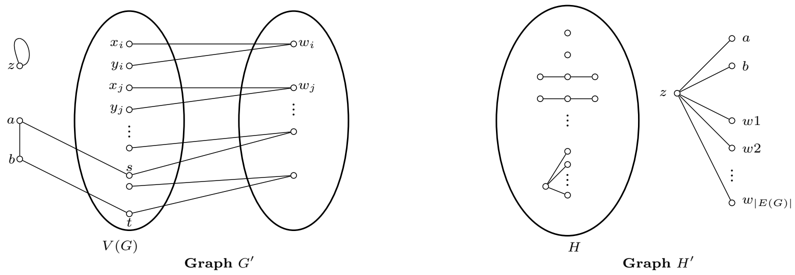

Figure 1 An illustration of construction of graph G' and H' in reduction from $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.

▶ **Theorem 19** ($\star$). *$\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT does not admit a polynomial compression unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Lower Bound for Kernel of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. In this subsection, we prove the main result of this section. We show that there does not exist a polynomial kernel of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Towards this we give a parameter preserving reduction from $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Given an instance (G, H, s, t, k) of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT, we construct an instance $(G', H', k+1)$ of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT as follows. Initially, we have $V(G') = V(H') = V(G) \cup \{z, a, b\}$. Now, for each edge $e_i \in E(G)$, add a vertex w_i to $V(G')$ and $V(H')$. Now, we define the edge set of G'. Let x_i, y_i be end points of $e_i \in E(G)$. For each $e_i \in E(G)$, add edges x_iw_i and y_iw_i to $E(G')$. Also, add a self loop on z in G' and edges sa, ab and bt to $E(G')$. To construct the edge set of H', we set $E(H') = E(H - \{s, t\})$. Additionally, we add zs, zt, za, zt, and zw_i for each $w_i \in V(H')$ to $E(H')$. Figure 1 describes the construction of G' and H'.

Clearly, H' belongs to $\mathcal{P}^{\star\star}_3$ and this construction can be carried out in the polynomial time. Now, we prove the equivalence between the instances (G, H, s, t, k) of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT and $(G', H', k+1)$ of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT in the following lemma.

▶ **Lemma 20.** *(G, H, s, t, k) is a yes-instance of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT if and only if $(G', H', k+1)$ is a yes-instance of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.*

Proof. In the forward direction, let (G, H, s, t, k) be a yes-instance of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT and S be one of its solution. We claim that $S \cup \{z\}$ is a solution to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT in $(G', H', k+1)$. In the graph G', since we subdivide each edge, all the paths from $s-t$ are of even length. Since, we subdivide each edge of G, $G' - \{a, b, z\}$ is a bipartite graph. Hence, an odd cycle in $G' - z$ consists of an $s-t$ path in $G' - \{a, b\}$ and edges sa, ab and bt. Clearly, by the construction of G', $(G' - \{a, b\}) \setminus S$ does not contain an $s-t$ path and hence $G' - z$ does not contain an odd cycle. Since, $H[S]$ is edgeless, $S \cup \{z\}$ is an independent set in H'. This completes the proof in the forward direction.

In the reverse direction, let S be a solution to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT in $(G', H', k+1)$. Since, $z \in S$, therefore, $s, t, a, b, w_i \notin S$ for any $w_i \in V(H')$. We claim that $S' = S \setminus \{z\}$ is a solution to $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT in (G, H, s, t, k). Suppose not, then there exists a $s-t$ path $(s, x_1, x_2, \cdots, x_l, t)$ in $G \setminus S'$. Correspondingly, there exists a $s-t$ path $(s, w_1, x_1, w_2, x_2, \cdots, x_l, w_{l+1}, t)$ in G' of even length which results into an odd cycle $(s, w_1, x_1, w_2, x_2, \cdots, x_l, w_{l+1}, t, b, a)$ in $G' \setminus S$, a contradiction. This completes the proof. ◀

Now, we present the main result of this section in the following theorem.

▶ **Theorem 21.** *$\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT does not admit a polynomial kernel. unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

6 Conclusion

In this paper we studied kernelization complexity of $\mathcal{D}_d$-CF-FVS and $\mathcal{D}_d$-CF-OCT. We showed that the former admits a polynomial kernel of size $k^{\mathcal{O}(d)}$, while $\mathcal{D}_d$-CF-OCT does not admit any polynomial kernel unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$. In fact, the later does not admit polynomial kernel even for much more specialized problem, namely $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Using much more involved marking scheme we can show that $\mathcal{D}_d$-CF-ECT admits polynomial kernel of size $k^{\mathcal{O}(d)}$. Similarly, we can extend the known polynomial kernel for OCT to CF-OCT when the conflict graph H has maximum degree at most one. Two most interesting questions that still remain open form our work are following: (a) does CF-FVS admit uniform polynomial kernel on graphs of bounded expansion; and (b) does CF-OCT admit a polynomial kernel when H is disjoint union of paths of length at most 2.

References

1. Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. Choice is Hard. In *ISAAC*, pages 318–328, 2015.
2. Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. Conflict-free Covering. In *CCCG*, pages 17–23, 2015.
3. Aritra Banik, Fahad Panolan, Venkatesh Raman, and Vibha Sahlot. Fréchet Distance Between a Line and Avatar Point Set. *FSTTCS*, pages 32:1–32:14, 2016.
4. Hans L. Bodlaender. On disjoint cycles. In *WG*, volume 570, pages 230–238, 1992.
5. Hans L. Bodlaender. A Cubic Kernel for Feedback Vertex Set. In *STACS*, volume 4393, pages 320–331, 2007.
6. Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *J. Discrete Math.*, 28:277–305, 2014.
7. Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The Undirected Feedback Vertex Set Problem Has a Poly(k) Kernel. In *IWPEC*, volume 4169, pages 192–202, 2006.
8. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
9. Andreas Darmann, Ulrich Pferschy, and Joachim Schauer. Determining a Minimum Spanning Tree with Disjunctive Constraints. In *ADT*, volume 5783, pages 414–423, 2009.
10. Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726–1735, 2011.
11. Rodney G. Downey and Michael R. Fellows. Fixed Parameter Tractability and Completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.
12. Leah Epstein, Lene M. Favrholdt, and Asaf Levin. Online variable-sized bin packing with conflicts. *Discrete Optimization*, 8(2):333–343, 2011.
13. Guy Even, Magnús M. Halldórsson, Lotem Kaplan, and Dana Ron. Scheduling with conflicts: online and offline algorithms. *J. Scheduling*, 12(2):199–224, 2009.
14. Pallavi Jain, Lawqueen Kanesh, and Pranabendu Misra. Conflict Free Version of Covering Problems on Graphs: Classical and Parameterized. *CSR*, pages 194–206, 2018.
15. Viggo Kann. Polynomially bounded minimization problems which are hard to approximate. In *ICALP*, pages 52–63, 1993.
16. Stefan Kratsch and Magnus Wahlström. Compression via Matroids: A Randomized Polynomial Kernel for Odd Cycle Transversal. *ACM Trans. Algorithms*, 10(4):20:1–20:15, 2014.

17 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering Small Independent Sets and Separators with Applications to Parameterized Algorithms. In *SODA*, pages 2785–2800, 2018.

18 David W. Matula and Leland L. Beck. Smallest-Last Ordering and clustering and Graph Coloring Algorithms. *J. ACM*, 30(3):417–427, 1983.

19 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On Parameterized Independent Feedback Vertex Set. *Theor. Comput. Sci.*, 461:65–75, 2012.

20 Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized Algorithms for Even Cycle Transversal. In *WG*, volume 7551, pages 172–183, 2012.

21 Ulrich Pferschy and Joachim Schauer. The Maximum Flow Problem with Conflict and Forcing Conditions. In *INOC*, volume 6701, pages 289–294, 2011.

22 Ulrich Pferschy and Joachim Schauer. The maximum flow problem with disjunctive constraints. *J. Comb. Optim.*, 26(1):109–119, 2013.

23 Ulrich Pferschy and Joachim Schauer. Approximation of knapsack problems with conflict and forcing graphs. *J. Comb. Optim.*, 33(4):1300–1323, 2017.

24 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

25 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010.