Parameterized Complexity of Conflict-Free Solutions

By

Lawqueen Kanesh

MATH10201504008

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

July, 2020

HOMI BHABHA NATIONAL INSTITUTE

Training School Complex, 2nd Floor, Anushaktinagar, Mumbai 400 094

Tel. : 022-25597627
Tele-Fax : 022-25503384

Ref.:- HBNI/RO/ 2018/ 87

31st January, 2019

CIRCULAR

In continuation to the circular dated 20th August 2018, regarding the "Guidelines on Promotion of Academic Integrity and Prevention of Plagiarism" in the Higher Educational Institutions as notified by the UGC, Govt. of India.

It was notified that the thesis will be accompanied by a Certificate signed by the student and duly endorsed by the Thesis Supervisor in a prescribed format.

It is henceforth informed that the students shall submit the thesis with a Certificate enclosing an undertaking along with the thesis document, that the thesis has been duly checked through a plagiarism detection tool. The PhD Supervisor shall also endorse the certificate indicating that the work done by the researcher under him/ her is plagiarism free.

The format for the undertaking/ Certificate given earlier shall be replaced with the one given in Annexure – I.

This shall come into force immediately.

B. Chandrasekar

Dr. B. Chandrasekar
Registrar

The Vice Chancellor, HBNI
All the Deans (Academics), CIs/OCC
Prof. D.K. Maity, Associate Dean, & Dean (Officiating), HBNI

Annexure I

CERTIFICATION ON ACADEMIC INTEGRITY

1. I Lavqueen Kanesh (name of the student) HBNI Enrolment No. MATH10201504008 hereby undertake that, the Thesis titled "***Parameterized Complexity of Conflict-Free Solutions***" ***(bold & italics)*** is prepared by me and is the original work undertaken by me and free of any plagiarism. That the document has been duly checked through a plagiarism detection tool and the document is plagiarism free.

2. I am aware and undertake that if plagiarism is detected in my thesis at any stage in future, suitable penalty will be imposed as per the applicable guidelines of the Institute / UGC.

10-02-2021

Signature of the Student
(with date)

Endorsed by the Thesis Supervisor:
(I certify that the work done by the Researcher is plagiarism free)

Signature (with date) 26/01/2024

Name: Saket Saurabh
Designation: Professor
Department/ Centre: Theoretical Computer Science,
Name of the CI/ OCC Institute of Mathematical Sciences

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by Lawqueen Kanesh entitled "Parameterized Complexity of Conflict-Free Solutions" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

Date: 01-December 2020

Chairman - Venkatesh Raman

Date: 01-December 2020

Guide/Convenor - Saket Saurabh

Date: 01-December 2020

Examiner - L. Sunil Chandran

Date: 01-December 2020

Member 1 - R. Ramanujam

Date: 01-December 2020

Member 2 - Geevarghese Philip

Date: 01-December 2020

Member 3 - Vikram Sharma

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: 01/12/2020

Place: Chennai

Saket Saurabh (Guide)

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Lawqueen Kanesh

DECLARATION

I hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Lawqueen Kanesh

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

1. **Conflict Free Version of Covering Problems on Graphs: Classical and Parameterized** : Pallavi Jain, Lawqueen Kanesh, Pranabendu Misra, *Theory of Computing Systems, 2020* (An extended abstract appeared in CSR 2018).

2. **Parameterized Complexity of Conflict-Free Matchings and Paths** : Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Saket Saurabh, *Algorithmica, 2020* (An extended abstract appeared in MFCS 2019).

Conferences

1. **Conflict Free Version of Covering Problems on Graphs: Classical and Parameterized** : Pallavi Jain, Lawqueen Kanesh, Pranabendu Misra, in the proceedings of *Computer Science - Theory and Applications - 13th International Computer Science Symposium in Russia, (CSR 2018).*

2. **Conflict Free Feedback Vertex Set: A Parameterized Dichotomy** : Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Daniel Lokshtanov, Saket Saurabh, in the proceedings of *43rd International Symposium on Mathematical Foundations of Computer Science, (MFCS 2018).*

3. **Exploring the Kernelization Borders for Hitting Cycles** : Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Pranabendu Misra, Saket Saurabh, in the proceed-

ings of *13th International Symposium on Parameterized and Exact Computation, (IPEC 2018).*

4. **Parameterized Complexity of Conflict-Free Matchings and Paths** : Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Saket Saurabh, in the proceedings of *44th International Symposium on Mathematical Foundations of Computer Science, (MFCS 2019).*

5. **Feedback Vertex Sets in Hypergraphs** : Pratibha Choudhary, Lawqueen Kanesh, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, in the proceedings of *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, (FSTTCS 2020).*

Lawqueen Kanesh

DEDICATIONS

This thesis is dedicated to my mother. Thank you for always believing in me.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Saket Saurabh. Without his guidance and utmost support, this journey would have been very difficult. He did not just teach me research but also helped me grow as a person. He always stood by me as a friend and a great advisor during academic and non-academic difficulties. He always inspired me and motivated me to never give up. I thank him with all my heart.

Besides my advisor, I thank my doctoral committee members and reviewers. I am grateful to the Theoretical Computer Science group at IMSc for their great teaching. I thank Prof. R. Ramanujam for mentoring me during my first year of Ph.D. I am also thankful to all the members of the IMSc staff, for all the support they provided.

I am thankful to all my co-authors—Akanksha, Pallavi, Daniel, Pranabendu, Fahad, and Pratibha. I would like to specially thank Akanksha, Pallavi, and Fahad for their patience with me during our long discussions. I thank Parameterized Complexity group at IMSc.

I want to thank all my dear friends at IMSc, without them, IMSc would not be the same. I want to specially thank Sanjukta, Oorna, Shivani, Aditi, Deeksha, Jayakrishnan, Abhishek, Pallavi, Akanksha, Roohani, and Pratik for their great friendship. I also want to thank my friends from IISc, MANIT, CTC, and Ainthinai.

Last but not least I am thankful to my parents, Resham Kanesh and Magan Singh Kanesh for their unconditional love and support. I thank them for providing me a great education and supporting my dreams. I am also thankful to my sister Vidhi and brother Vidhan for their love and friendship. I am forever indebted to my family for their immense love and care.

Contents

List of Figures

Part IV

Conclusion

Chapter 8

Conclusion and Open Problems

In this thesis we studied two generalizations (variants) of classical problems in the Graph Algorithms and Parameterized Complexity. In the first part of the thesis, we introduced and studied conflict-free variant of some of the classical problems, and in the second part, we extended the FVS problem to Hypergraphs. We explored both the variants in the realm of Parameterized complexity and obtained various algorithmic and hardness results.

Our work on conflict-free variant of classical problems opens up a whole new area of research in obtaining dichotomy results. For every property Π, where CF-Π-VD is W[1]-hard, it is a natural question to ask for which families of graphs G, H does the problem becomes FPT. In particular, for which graph classes $\mathscr{G}, \mathscr{H}$, the problem $(\mathscr{G}, \mathscr{H})$-CF-$\mathscr{Q}$ admit FPT algorithms and polynomial kernels, where $\mathscr{Q}$ could be any classical problem in Graph Algorithms and Parameterized Complexity. Two most interesting questions that still remain open for CF-FVS and CF-OCT 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. For properties Π with finite forbidden characterization one direction of research is to obtain faster running time algorithms. Another interesting question is to obtain (parameterized) dichotomy results for CF-MM and CF-SP, based on the families of graphs where the

input graphs belong to. Another direction could be studying kernelization complexity for different families of graphs, and also to see what all FPT problems remain FPT with the conflicting constraints.

FVS on Hypergraphs: In the second part of the thesis, we initiated the study of FEEDBACK VERTEX SET problem on hypergraphs. We showed that the problem is W[2]-hard on general hypergraphs and admits FPT algorithms when the input is restricted to d-hypergraphs, and linear hypergraphs. We believe that this opens up a new direction in the study of parameterized algorithms. That is, extending the study of other graph problems, in the realm of Parameterized Complexity, to hypergraphs. Designing substantially faster algorithms for HFVS on linear hypergraphs and designing polynomial kernels remain interesting questions for the future.

Summary

The thesis is divided into two parts. In the first part of the thesis we introduce a new variant for some of the classical problems in Graph Algorithms, we call it *conflict-free version*. We study them from the viewpoint of classical and Parameterized Complexity. In the second part of the thesis we extend the FEEDBACK VERTEX SET (FVS) problem to hypergraphs and study from the view point of Parameterized Complexity.

We begin by studying conflict-free versions of vertex deletion problems. Let Π be a family of graphs (or property) – such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs or planar graphs. In the vertex deletion problem corresponding to Π (Π-VD), given a graph G, and a non-negative integer k, the goal is to delete a set S of at most k vertices such that $G-S$ is in Π. In the conflict-free version of vertex deletion problem corresponding to Π CF-Π-VD, we are given a conflict graph H together with the graph G and integer k, and the goal is to find a set $S \subseteq V(G)$ of size at most k, such that S is a solution to (G,k) of Π-VD and S is an independent set in H. We study the complexity of CF-Π-VD based on the forbidden set characterization of the property Π. For graph properties with finite forbidden characterization, we show that CF-Π-VD is FPT and admits a polynomial kernel. For graph properties without finite forbidden characterization, we show that if Π is characterized by a "well-behaved" infinite family of forbidden induced subgraphs, then CF-Π-VD is W[1]-hard. In particular, we show that conflict-free version of FVS (CF-FVS) is W[1]-hard even when G is disjoint union of cycles. A similar result holds

for conflict-free version of ODD CYCLE TRANSVERSAL (CF-OCT), CHORDAL VERTEX DELETION (CF-CVD) and INTERVAL VERTEX DELETION (CF-IVD). We also show that, CF-FVS, CF-OCT, CF-CVD, and CF-IVD are FPT, when H belongs to the family of d-degenerate graphs or nowhere dense graphs. For this purpose we use the notion of "*k-independence covering family*" introduced in [81]. We obtain a complete dichotomy result on the Parameterized Complexity of the problem $\mathscr{H}$-CF-FVS, where the conflict-free graph H belongs to graph class $\mathscr{H}$ (for hereditary $\mathscr{H}$), in terms of the INDEPENDENT SET problem. We show that $\mathscr{H}$-CF-FVS is in FPT if and only if $\mathscr{H}$+CLUSTER IS is in FPT, where $\mathscr{H}$+CLUSTER IS is the INDEPENDENT SET problem on the edge union graph of a cluster graph and a graph in $\mathscr{H}$.

We obtain a polynomial kernel for CF-FVS and show that CF-OCT does not admit a polynomial kernel, when H belongs to the family of d-degenerate graphs. We also study conflict-free (parameterized) versions of the MAXIMUM MATCHING (CF-MM) and SHORTEST PATH (CF-SP) problems. We show that both CF-MM and CF-SP are W[1]-hard, when parameterized by the solution size. For the CF-MM problem, we give an FPT algorithm, when the conflict graph belongs to the family of chordal graphs. For conflict graphs being d-degenerate and nowhere dense graphs, we obtain FPT algorithms for both CF-MM and CF-SP. We study a variant of CF-MM and CF-SP, where instead of conflicting conditions being imposed by independent sets in a conflict graph, they are imposed by independence constraints in a (representable) matroid. We give FPT algorithms for the above variant of both CF-MM and CF-SP.

Finally, we extend our study from problems on graphs to hypergraphs. In particular, we study the FEEDBACK VERTEX SET problem on hypergraphs. We show that FVS on hypergraphs is W[2]-hard, when parameterized by the solution size. We obtain FPT algorithms for FVS on hypergraphs, when the input hypergraph is restricted to d-hypergraphs and *linear hypergraphs*.

Part I

Introduction

Chapter 1

Introduction

Graphs are mathematical structure that model pairwise relations between various objects. Vertices of a graph represents objects and edges represents relations between vertices. Graph theory is the field that study graphs and their properties. It came into light long back in the early seventeenth century. In 1736 Leonhard Euler wrote a paper on, the historically remarkable problem, Seven Bridges of Königsberg [12], which is considered to be the first paper in Graph Theory. On the other hand, the class of algorithms that study computational complexity of problem arising in Graph Theory is Graph Algorithms. A major sub-field in Graph Algorithms are graph-modification problems, where the goal is to delete a set of vertices, or a set of edges, or add a set of edges of minimum or maximum size such that the resulting graph satisfies certain properties or becomes a member of some well-understood graph class. However, most of these problems are NP-complete [111, 76]. Apart from these NP-hard problem there are also many polynomial time solvable problems that are of major importance and have been extensively studied in Graph Algorithms. Some examples are MINIMUM CUT, MAXIMUM MATCHING and SHORTEST PATH. To cope with NP-completeness [54, 84, 51], it is a common practice to find efficient algorithms for restricted classes of inputs, or find approximation algorithms, or explore Parameterized Complexity of the problem. In this thesis we introduce a new variant and generalization of

some of the above mentioned classical problems. We call it *conflict-free version*, where together with the input of the classical problem, we are also given pairs which are in conflicts (impossible pairs), that is, a pair cannot go into a solution together if they are in conflict. We present the conflict constraints using graphs or matroids. We study the classical and Parameterized Complexity of these problems. When we are not interested in the exact value of the polynomial factor in the running time, then we use the $\mathcal{O}^\star$ -notation, which suppresses factors polynomial in the input size.

One of the graph modification problem that is studied more extensively than others in the thesis is FEEDBACK VERTEX SET. In the parameterized FEEDBACK VERTEX SET (FVS) problem, given a graph G, and an integer k, the goal is to find a set $S \subseteq V(G)$ of size at most k such that S hits all the cycles in G, that is, after deleting the set S from G, the remaining graph $G-S$ is a forest (acyclic). The optimization version of FVS is one of the Karp's 21 NP-complete problems and it is NP-hard even on the graphs with maximum degree at most four. It remains NP-complete on directed graphs (edges have orientation). We refer to the problem in directed graphs as DIRECTED FEEDBACK VERTEX SET. In the parameterized DIRECTED FEEDBACK VERTEX SET problem, given a directed graph D, and an integer k, the goal is to find a set $S \subseteq V(G)$ of size at most most k such that S hits all the directed cycles in D, that is, after deleting the set S from D, the remaining graph $D-S$ is a directed acyclic graph. FVS is a central problem in the ream of Parameterized Complexity [29, 23, 26, 31, 59, 70, 77]. This problem is known to be in FPT, and the fastest known (randomized) algorithm for it runs in time $\mathcal{O}^\star(2.7^k)$ for undirected graphs [77]. For directed graphs Chen et al. gave a $\mathcal{O}(4^k k! k^4 nm)$ running time algorithm [26] and Lokshtanov et al gave a $\mathcal{O}(4^k k! k^5 (n+m))$ time algorithm [83], which are currenlty the fastest known algorithms. Several variant and generalizations of FEEDBACK VERTEX SET and DIRECTED FEEDBACK VERTEX SET such as WEIGHTED FEEDBACK VERTEX SET [3, 26], INDEPENDENT FEEDBACK VERTEX SET [2, 90], CONNECTED FEEDBACK VERTEX SET [91], SIMULTANEOUS FEEDBACK VERTEX SET [4, 20], and DIRECTED SUBSET FEEDBACK VERTEX SET [28]

have been studied from the viewpoint of Parameterized Complexity. FVS and DIRECTED FEEDBACK VERTEX SET on restricted classes of inputs have also been vastly studied in literature. In the first part of the thesis we study conflict-free variants of FVS and DIRECTED FEEDBACK VERTEX SET. We also study them on some restricted classes of inputs such as graphs of bounded degeneracy, nowhere dense graphs, and tournaments. In the second part of the thesis we study a generalization of FVS in hypergraphs and also study it on some restricted classes of hypergraphs.

A *parameterized problem* Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet set. An instance of a parameterized problem is a tuple (x,k), where x is a classical problem instance and k is an integer, called the *parameter*. A central notion in Parameterized Complexity is *fixed-parameter tractability* (or in FPT) which means, for a given instance (x,k), decidability in time $f(k) \cdot \mathsf{poly}(|x|)$, where $f(\cdot)$ is an arbitrary computable function and $\mathsf{poly}(\cdot)$ is a polynomial function. To prove that a problem is FPT, it is possible to give an explicit algorithm, called a *parameterized algorithm*, which solves it in time $f(k) \cdot \mathsf{poly}(|x|)$. On the other hand, to show that a problem is unlikely to be in FPT, it is possible to use FPT time reductions analogous to the polynomial time reductions employed in classical complexity. Here, the concept of W[t]-hardness replaces the concept of NP-hardness, and we need not only construct an equivalent instance in FPT time, but also ensure that the size of the parameter in the new instance depends only on the size of the parameter in the original instance. For more details on Parameterized Complexity, we refer the reader to the books of Downey and Fellows [44], Flum and Grohe [49], Niedermeier [96], and the recent book by Cygan et al. [29].

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 cannot 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. A 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.

1.1 Conflict-Free Problems

In the past, the conflict-free versions of some classical problems have been studied, e.g. for SHORTEST PATH [66], MAXIMUM FLOW [98, 99], KNAPSACK [100], BIN PACKING [45], SCHEDULING [46], MAXIMUM MATCHING and MINIMUM WEIGHT SPANNING TREE [36, 35]. 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 [5, 6, 7]).

Motivated by these works, we studied the conflict-free versions of several classical problems in Graph Theory and Parameterized Complexity. In any classical problem $\mathcal{Q}$, given an input I, the goal is to output a set S which satisfy certain properties specific to the problem $\mathcal{Q}$. In the conflict-free version of $\mathcal{Q}$, namely CONFLICT-FREE Q (CF-$\mathcal{Q}$, in short), together with the input I, we are also given a conflict graph H, and we require that the output set S satisfy two constraints. First, S is a solution to $\mathcal{Q}$, and second, S is an independent set in the graph H.

We initiated the study of the conflict-free versions of several well studied vertex deletion problems in Parameterized Complexity. A typical parameterized vertex deletion problem on graphs is of the following form. Let Π be a family of graphs (or property) – such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs or planar graphs. The vertex deletion problem corresponding to Π is formally

stated as follows.

Π-VERTEX DELETION	**Parameter:** k
Input: An undirected graph G and a non-negative integer k.	
Question: Does there exist $S \subseteq V(G)$, such that $\|S\| \leq k$ and $G - S$ is in Π?	

That is, given a graph G, can we delete at most k vertices such that the resulting graph belongs to Π? The set S ***is called a*** Π***-deletion set***. The conflict-free vertex deletion problem corresponding to Π is formally stated as follows.

CONFLICT-FREE Π-VERTEX DELETION (CF-Π-VD)	**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 a set $S \subseteq V(G)$, such that $\|S\| \leq k$, $G - S$ is in Π and S is an independent set in H?	

We define CF-Π-VD, when both the graphs G, H are directed graphs (hypergraphs), appropriately. Where, the notion of independent set in directed graph is similar to that of undirected graphs. In hypergraphs, a set S of vertices in a hypergraph G is called independent if no two vertices in S are contained in an edge in G. Observe that when H is the edgeless graph, CF-Π-VD is the same as Π-VERTEX DELETION and thus it generalizes the non-conflict-free version of the problem. Furthermore, when H is the same as G it corresponds to *independent* version of these problems which are also well studied, such as INDEPENDENT FEEDBACK VERTEX SET [90, 81]. Thus, CF-Π-VD is a generalization of well studied problems in algorithms and complexity.

A *graph property* Π is a set of graphs, and a graph in Π is called a Π-*graph*. We say that Π is *hereditary* if for any graph G in Π, every induced subgraph of G is also in Π. A graph property Π has a forbidden set characterization if there is a set $\mathscr{F}$ of graphs such that a graph is a Π-graph if and only if it does not contain any graph in $\mathscr{F}$ as an induced subgraph, and further, it has a finite forbidden characterization if $\mathscr{F}$ is a finite set. We study the complexity of CF-Π-VD based on the forbidden set characterization of the property

Π.

1.1.1 Properties With Finite Forbidden Characterization

The problem CF-Π-VD is FPT whenever Π has a finite forbidden set characterization. Indeed, this problem admits an algorithm with running time $\mathcal{O}^\star(\alpha^k)$, where α is the size of a largest graph in the finite forbidden subgraphs in Π and k is the parameter. Furthermore, it also admits a polynomial vertex kernel. See Chapter 3 for these results. For several well-studied cases of Π-VERTEX DELETION, where Π is characterized by the finite family of forbidden induced subgraphs we can obtained algorithms with improved running time than the generic result stated above. We studied conflict-free versions of VERTEX COVER, d-HITTING SET, SPLIT VERTEX DELETION and FEEDBACK VERTEX SET IN TOURNAMENTS problems and obtained improved results.

1.1.1.1 VERTEX COVER

When Π is the family of edgeless graphs, then the Π-VERTEX DELETION is known as VERTEX COVER (VC). It is one of the Karp's 21 NP-complete problems and is a very well studied problem in Parameterized Complexity [27]. The fastest known algorithm for VC runs in time $\mathcal{O}^\star(1.2738^k)$ [27]. We studied its conflict-free version namely CONFLICT FREE VERTEX COVER (CF-VC), where given two graphs G, H and an integer k, the goal is to find a set $S \subseteq V(G)$ such that $G - S$ is edgeless and S is an independent set in H. While VC is polynomial time solvable on graphs of degree at most 2, the CF-VC problem is NP-complete even when the graph G is of degree at most 2. This holds even when G is the disjoint union of P_3s (P_ℓ denotes the path on ℓ vertices). CF-VC is polynomial time solvable when G has degree at most one, or when both G and H have a perfect matching. Analogous to VC, the CF-VC problem admits a $2k$-vertex kernel, a factor 2-approximation algorithm and an $\mathcal{O}^\star(1.1996^n)$ exact algorithm. It is easy to obtain a $\mathcal{O}^\star(2^k)$ FPT algorithm for CF-VC as the forbidden subgraph is an edge. However, CF-VC

admits a faster FPT algorithm of running time $\mathcal{O}^\star(1.2738^k)$. These results can be found in Chapter 3.

1.1.1.2 d-HITTING SET

HITTING SET problem can be viewed as VC in hypergraphs. In HITTING SET, given a universe U and a family $\mathcal{F}$ of subsets of U as input, the aim is to find a set $S \subseteq U$ of size at most k such that S intersects with every set in $\mathcal{F}$. When the size of each set in $\mathcal{F}$ is bounded by d then the problem is known as d-HITTING SET. It is well known that the HITTING SET problem is W[2]-complete, when parameterized by solution size, while the d-HITTING SET problem is FPT parameterized by the solution size and d [1, 49]. The fastest running algorithm known for d-HITTING SET to the best of our knowledge runs in time $\mathcal{O}^\star((d-0.7262)^k)$ [107]. In this thesis we study the conflict-free version of d-HITTING SET, namely CONFLICT FREE d-HITTING SET (d-CF-HS). It is easy to see that by exploiting the forbidden subgraph characterization of d-CF-HS, we immediately get a $\mathcal{O}^\star(d^k)$ FPT algorithm for d-CF-HS. We use the algorithm in [107] as subroutine and obtain a faster algorithm for d-CF-HS running in $\mathcal{O}^\star(((d-1)+0.2738)^k) = \mathcal{O}^\star((d-0.7262)^k)$ time. These results can be found in Chapter 3.

1.1.1.3 SPLIT VERTEX DELETION

An undirected graph is split graph if its vertex set can be partitioned into two sets, where one set induces a complete graph as a subgraph and the other set induces an independent set (edgless graph) as subgraph. The forbidden induced subgraph for split graphs are a cycle on four vertices (C_4) , a cycle on five vertices (C_5), and a pair of disjoint edges ($2K_2$) [50]. The problem deletion to split graph is known as SPLIT VERTEX DELETION (SVD) and the fastest known algorithm for SVD to the best of our knowledge runs in time $\mathcal{O}^\star(1.2738^k k^{\mathcal{O}(\log k)})$ [32]. In this thesis, we study the conflict-free version of SVD, namely CONFLICT FREE SPLIT VERTEX DELETION (CF-SVD). It is easy to see that

by exploiting the forbidden subgraph characterization of CF-SVD, we immediately get a $\mathcal{O}^\star(5^k)$ time FPT algorithm for CF-SVD. We use approach similar to [32] and obtain a faster algorithm for CF-SVD running in $\mathcal{O}^\star(1.2738^k k^{\mathcal{O}(\log k)})$ time and polynomial space. These results can be found in Chapter 3.

1.1.1.4 FEEDBACK VERTEX SET IN TOURNAMENTS

A directed graph is tournamnet if every pair of vertices is connencted by a directed edge.CONFLICT FREE DIRECTED FEEDBACK VERTEX SET on tournaments is very well studied in the ream of Parameterized Complexity [42, 74]. The problem is refereed as FEEDBACK VERTEX SET IN TOURNAMENTS (FVST). The forbidden induced subgraph for tournaments is a directed cycle on three vertices (triangle). The fastest known algorithm for FVST to the best of our knowledge runs in time $\mathcal{O}^\star(1.618^k)$ [74]. In this thesis, we study the conflict-free version of FVST, namely CONFLICT FREE FEEDBACK VERTEX SET IN TOURNAMENTS (CF-FVST). It is easy to see that by exploiting the forbidden subgraph characterization of CF-FVST, we immediately get a $\mathcal{O}^\star(3^k)$ time FPT algorithm for CF-FVST. We obtain a faster algorithm for CONFLICT FREE FEEDBACK VERTEX SET IN TOURNAMENTS running in $\mathcal{O}^\star(2^k)$ time using iterative compression technique. These results can be found in Chapter 3.

1.1.2 Properties Without finite Forbidden Characterization

For graph properties that are not characterized by a finite family of forbidden induced subgraphs and if Π is characterized by a "well-behaved" infinite family of forbidden induced subgraphs, then CF-Π-VD is W[1]-hard. This motivates to restrict input graph classes. In this thesis we consider conflict-free variants of FEEDBACK VERTEX SET, ODD CYCLE TRANSVERSAL, CHORDAL VERTEX DELETION and INTERVAL VERTEX DELETION problems. Note that all these problems do not admit finite forbidden characterization. We explore their Parameterized Complexity for some restricted input graphs. Mainly

we study classes of bounded degeneracy graphs and nowhere dense graphs. We use the computation of an *independence covering family*, a notion which was recently introduced by Lokshtanov et al. [81] to design FPT algorithms for these classes (See Chapter 3). It is worth noting that these classes include trees, graphs of bounded degree, planar graphs, graphs that exclude a fixed graph as a minor (or a topological minor) and graphs of bounded expansion.

1.1.2.1 FEEDBACK VERTEX SET

We refer to the conflict-free variant of FVS as CF-FVS. We studied CF-FVS when the conflict graph is restricted to a graph class $\mathscr{F}$. We call it as $\mathscr{F}$-CF-FEEDBACK VERTEX SET ($\mathscr{F}$-CF-FVS, for short). In contrast to FVS, $\mathscr{F}$-CF-FVS is W[1]-hard on general graphs and admits an FPT algorithm if $\mathscr{F}$ is the family of graphs of bounded degeneracy or nowhere dense graphs. See Chapter 3 for these results. $\mathscr{F}$-CF-FVS admits polynomial kernel when $\mathscr{F}$ is the family of graphs of bounded degeneracy. The main tool that we use to obtain polynomial kernel is a combinatorial tool of *k-independence preserver*. Informally, given a graph G, a set $X \subseteq V(G)$, and an integer k, it is a set of *important* vertices that is enough to capture the independent set property in G. We show that for d-degenerate graph independence preserver of size $k^{\mathcal{O}(d)}$ exists, and can be used in designing polynomial kernel. We believe that the tool can be useful for several other combinatorial problems. See Chapter 5 for these result. The problem $\mathscr{F}$-CF-FVS can be related to the INDEPENDENT SET problem on special classes of graphs, which gives a complete dichotomy result on the Parameterized Complexity of the problem $\mathscr{F}$-CF-FVS, when $\mathscr{F}$ is a hereditary graph family. In particular, $\mathscr{F}$-CF-FVS is FPT parameterized by the solution size if and only if $\mathscr{F}+$CLUSTER IS is FPT parameterized by the solution size. Here, $\mathscr{F}+$CLUSTER IS is the INDEPENDENT SET problem in the (edge) union of a graph $G \in \mathscr{F}$ and a cluster graph H (G and H are explicitly given). The problem $\mathscr{F}+$CLUSTER IS is FPT when $\mathscr{F}$ is the family of $K_{i,j}$-free graphs. For the family of bipartite graph $\mathscr{B}$, $\mathscr{B}$-CF-FVS is W[1]-hard, when parameterized by the solution size. For the family of graphs $\mathscr{F}_\varepsilon$, which comprise of

graphs G such that $|E(G)| \leq |V(G)|^{2-\varepsilon}$, for each $0 < \varepsilon < 1$, $\mathscr{F}_\varepsilon$-CF-FVS is W[1]-hard, when parameterized by the solution size, for every $0 < \varepsilon < 1$. These results can be found in Chapter 4.

1.1.2.2 ODD CYCLE TRANSVERSAL and s-t CUT

In the ODD CYCLE TRANSVERSAL problem, given a graph G and an integer k, the aim is to find a subset S of vertices of G of size at most k such that S hits every cycle of odd length in G, that is, the graph $G-S$ is bipartite. The paramaterized complexity of ODD CYCLE TRANSVERSAL was a major open problem for a long time, until Reed et. al gave an FPT algorithm in 2003 [102] and later faster algorithms were designed [68, 80, 62, 63]. Kratsch and Wahlström [71] gave a randomized polynomial kernel for ODD CYCLE TRANSVERSAL using matroid theory. In contrast to ODD CYCLE TRANSVERSAL, the directed variant (DIRECTED ODD CYCLE TRANSVERSAL) is W[1]-hard parameterized by solution size. In this thesis we study conflict-free variant of ODD CYCLE TRANSVERSAL, namely CONFLICT FREE ODD CYCLE TRANSVERSAL (CF-OCT). Unlike, ODD CYCLE TRANSVERSAL, the problem CONFLICT FREE ODD CYCLE TRANSVERSAL is W[1]-hard on general graphs. It admits an FPT algorithm when the conflict graph belongs to the family of graphs of bounded degeneracy or nowhere dense graphs. See Chapter 3 for these results. CF-OCT does not admit polynomial kernel even when the conflict graph belongs to the class of disjoint union of paths of length at most three and at most two star graphs. To study kernelization complexity of CF-OCT, we study conflict-free variant of the s-t-CUT problem, namely CF-s-t-CUT. In the s-t-CUT problem, given a graph G and two vertices s,t, the aim is to find a set S of minimum size such that S hits every s to t path in G, that is, s,t are not reachable from each other in the graph $G-S$. It can be solved in polynomial time [34, 40]. In contrast to the classical version, the CF-s-t-CUT problem is NP-hard even when the conflict graph belongs to the class of disjoint union of paths of length at most three. To show no polynomail kernel for CONFLICT FREE ODD CYCLE TRANSVERSAL, we use the technique of cross-composition using CF-s-t-CUT.

See Chapter 5 for these results.

1.1.2.3 CHORDAL VERTEX DELETION and INTERVAL VERTEX DELETION

A graph is chordal if it does not contain any induced cycle on at least 4 vertices, that is, every cycle of length four or more vertices have a chord. The class of interval graphs is a proper subset of chordal graphs. A graph is interval graph if it is chordal and does not contain any asteroidal triple (a set of three vertices such that each pair of them is connected by a path which avoids the neighbors of the third one). In the CHORDAL VERTEX DELETION (INTERVAL VERTEX DELETION) problem, given a graph G, the aim is to find a subset S of vertices of G of size at most k such that the graph $G-S$ is chordal (interval). The problem CHORDAL VERTEX DELETION and INTERVAL VERTEX DELETION are known to be in FPT [25, 21]. In this thesis we study their conflict-free variants. In contrast to their classical versions the conflict-free variants of CHORDAL VERTEX DELETION and INTERVAL VERTEX DELETION are W[1]-hard on general graphs. They admit FPT algorithms when the conflict graph belongs to the family of graphs of bounded degeneracy or nowhere dense graphs. These results can be found in Chapter 3.

1.1.3 SHORTEST PATH and MAXIMUM MATCHING

MAXIMUM MATCHING and SHORTEST PATH are among the classical graph problems which are extensively studied for theoretical and practical purposes in Graph Theory. In the MAXIMUM MATCHING problem, give a graph G, the goal is to compute a maximum sized subset X of edges such that no two edges in X have a common vertex. This problem is known to be solvable in polynomial time [47, 88]. In the SHORTEST PATH problem, given a graph G and two vertices s,t, the goal is to compute a path of minimum length (minimum number of vertices) between s and t. The SHORTEST PATH problem, together with its variants such as all-pair shortest path, single-source shortest path, weighted shortest path, etc. are known to be solvable in polynomial time [41, 9].

Darmann et al. [36] (among other problems) studied the conflict-free variants of MAXIMUM MATCHING and SHORTEST PATH and showed that the conflict-free variant of MAXIMUM MATCHING is NP-hard even when the conflict graph is a disjoint union of edges (matching). Moreover, for the conflict-free variant of SHORTEST PATH, they showed that the problem is APX-hard, even when the conflict graph belongs to the family of 2-ladders. In this thesis we study the conflict-free variants of MAXIMUM MATCHING and SHORTEST PATH, called as CF-MM and CF-SP, respectively, from the view point of paramterized complexity. We show that both CF-MM and CF-SP are W[1]-hard, when parameterized by the solution size. CF-MM is W[1]-hard even when the graph where we want to compute a matching is itself a matching and CF-SP is W[1]-hard even when the conflict graph belongs to the class of unit interval graphs. Both the problems are FPT when the conflict graph belongs to the class of bounded degeneracy graphs and nowhere dense graphs. While CF-MM admit FPT algorithm, when the conflict graph belongs to the family of chordal graphs, CF-SP problem is W[1]-hard even when the conflict graph belongs to the class of unit-interval graphs. We also study CF-MM and CF-SP, when instead of conflicting conditions being imposed by independent sets in a conflict graph, they are imposed by independence constraints in a (representable) matroid. Both the problems admit FPT algorithms for the above variant. These results can be found in Chapter 6.

1.2 Feedback Vertex Set on Hypergraphs

Hypergraphs are a generalization of graphs, where an edge can connect more than two vertices. While in graphs each edge contains exactly two vertices, in hypergrapgs each edge is a subset of vertices. Hypergraphs are essentially a set family H: we have a universe $V(H)$ and a family of hyperedges $E(H)$, where each hyperedge (or an edge) is a subset of $V(H)$. When every hyperedge in $E(H)$ is of size at most d, it is known as a d-hypergraph. Extension of concepts and ideas from Graph Theory to Hypergraph Theory is widely been

studied in recent decades [17, 108]. In fact it is a folklore that the theory of hypergraphs was invented for generalizing Graph Theory. The problem VERTEX COVER is extended as HITTING SET in Gypergraph Theory and for d-hypergraphs the problems is known as d-HITTING SET, which are extensively studied [107]. However, there is almost no study of FVS on hypergraphs. The only known result is a factor d approximation for FVS on d-hypergraphs [55]. To fill this gap, in this thesis we study hypergraph variant of the FVS problem from the view point of Parameterized Complexity

One of the main reasons for the lack of study of FVS on hypergraphs is that it is not quite as natural to define the generalization of FVS in hypergraphs, as it is for the case of VC (generalizing to HITTING SET and d-HITTING SET) in hypergraphs. To generalize the notion of FVS to hypergraphs, we need to have notions of *cycles* and *forests* in hypergraphs. For cycles, we use the same notion as that in Graph Theory [39]: a cycle in a hypergraph H is a sequence $(v_0, e_0, v_1, \ldots, v_\ell, e_\ell, v_0)$ such that $v_0, \ldots, v_\ell$ are distinct vertices, $e_0, \ldots, e_\ell$ are distinct hyperedges, $\ell \geq 1$ and $v_i, v_{(i+1) \mod (\ell+1)} \in e_i$ for any $i \in \{0, \ldots, \ell\}$. Given the above definition of cycle, a subset S of vertices in a hypergraph H is called a *feedback vertex set*, if there does not exist a cycle in the hypergraph obtained after *deleting* vertices in S. The next natural question is what do we mean by *deletion* of a vertex in a hypergraph. Again, there are two natural ways to define the vertex deletion operation in hypergraphs:

1. One way to define the operation of deletion of a vertex v is to delete the vertex v and all the hyperedges containing the vertex v – this is termed as *strong deletion* or simply *deletion*.
2. Other way is to delete only the vertex v, without deleting the hyperedges that contain v; this is termed as *weak deletion*. That is, the hypergraph H' obtained after weak deletion of a vertex v from H has vertex set $V(H)$ and edge set $\{e \in E(H) \colon v \notin e\} \cup \{e \setminus \{v\} \colon e \in E(H), v \in e, |e| > 2\}$.

Observe that both the notions of cycles and deletion of vertices naturally generalize similar notions from graphs. For a hypergraph H we use the notation $H - S$ to denote the graph

obtained after (strong) deletion of the vertices in S. Consequently, there are two ways one may define the FVS problem – WEAK FVS and STRONG FVS. The idea of studying STRONG FVS on hypergraphs is our main conceptual contribution.

Given a hypergraph H, the incidence bipartite graph G corresponding to H is a bipartite graph with bipartition $V(G) = A \uplus B$ where $A = V(H)$ and $B = E(H)$, and for any $v \in V(H)$ and $e \in E(H)$, ve is an edge in G if $v \in e$ in H. Observe that WEAK FVS corresponds to finding a fvs S of size at most k, such that $S \subseteq A$ and $G - S$ is a forest. Thus, using the best known algorithm for WEIGHTED FVS we can solve WEAK FVS by transforming the problem to WEIGHTED FVS. To transform WEAK FVS to WEIGHTED FVS we assign every vertex in B a weight of $k+1$, every vertex in A a weight of 1. Now the problem of finding an fvs of weight at most k will be equivalent to solving WEAK FVS for the original hypergraph. We consider FVS on hypergraphs with respect to strong deletion. FVS on general hypergraphs is W[2]-hard when parameterized by k, which is not surprising as HITTING SET is W[2]-hard. We study the problem for the cases when the input is restricted to d-hypergraphs and *linear hypergraphs*. A hypergraph H is linear if $|e \cap e'| \leq 1$ for any two distinct hyperedges $e, e' \in E(H)$. For both these families FVS admits FPT algorithms. Our main result is a randomized algorithm for the case when the input hypergraph is linear, and the size of the hyperedges is not bounded. These results can be found in Chapter 7.

Chapter 2

Preliminaries

2.1 Graphs and sets:

Throughout the thesis, we use the following notions. For the notations related to graphs that are not explicitly stated here, we refer to the book [40].

Sets: We denote the set of natural numbers, real numbers and integers by $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{Z}$, respectively. For $t \in \mathbb{N}$, by $[t]$ and $[0,t]$, we denote the sets $\{1,2,\cdots,t\}$ and $\{0,1,2,\cdots,t\}$, respectively. For two sets X and Y, by $X \setminus Y$ we mean the set $\{v | v \in X, v \notin Y\}$.

Graphs: For a (directed) graph G, we use $V(G)$ to denote the vertex set and $E(G)$ to denote the (arc) edge set of the (directed) graph G. When the graph is clear from the context, we use n and m to denote the number of vertices and edges in the graph, respectively. Let G and H be graphs, $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, then we say that H is a *subgraph* of G. For a graph G and a set $X \subseteq V(G)$, by $G[X]$ we mean the graph G induced on X, that is, the subgraph of G with vertex set X and edge set $\{uv \in E(G) \mid u,v \in X\}$. Moreover, by $G-X$ we mean the graph $G[V(G) \setminus X]$, that is, the graph with the vertex set $V(G)$ and the edge set $E(G) \setminus X$. Let E' be a subset of edges of the graph G, by $G[E']$ we mean the graph with

vertex set $V(G)$ and edge set E'. For a graph G and a subset of vertices $U \subseteq V(G)$, $N_G(U)$ and $N_G[U]$ denote the open neighborhood and closed neighborhood of U, respectively. That is, $N_G(U) = \{v \in V(G) \colon u \in U \text{ and } uv \in E(G)\} \setminus U$ and $N_G[U] = N_G(U) \cup U$. If $U = \{u\}$, then we write $N_G(u) = N_G(U)$ and $N_G[u] = N_G[U]$. In particular, for $u \in V(G)$, $N_G(u)$ denotes the set of vertices which have at least one edge incident on u in G. 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 $\deg_G(v)$, that is, $\deg_G(v)$ denotes the size of $N_G(v)$. The subscript from the notations are ommited when it is clear from the context.

For $v_1, v_\ell \in V(G)$, a *v_1v_ℓ-path* $P = (v_1, v_2, \cdots, v_{\ell-1}, v_\ell)$ in G is a sequence of (distinct) vertices, such that the set of vertices in P ($V(P)$) is a subset of $V(G)$ and for each $i \in [\ell - 1]$, we have $v_iv_{i+1} \in E(G)$. Moreover, the edges in $E(P) = \{v_iv_{i+1} \mid i \in [\ell - 1]\}$ are called set of edges in P and v_1, v_n are endpoints of P. The *length* of a path is the number of edges in it. A *shortest uv-path* is a uv-path with minimum number of edges. A *cycle* $C = (v_1, \ldots, v_n)$ is a path with the edge v_1v_n. Consider graphs G and H. We say that G is an *H-free* graph if no subgraph of G is isomorphic to H. We define a *degree two induced path in G* or a *degree two path* 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 be a graph. An *independent set* in G is a set $X \subseteq V(G)$ such that for every $u, v \in X$, $uv \notin E(G)$. A *matching* in G is a set $Y \subseteq E(G)$ such that no two distinct edges in Y have a common vertex. A matching M in G is a *maximum matching* if for any matching Y in G, $|M| \geq |Y|$. A matching M in G saturates a set $X \subseteq V(G)$, if every vertex in X is an end point of an edge in M.

A *triangle* is a cycle consisting of exactly 3 edges. A *chordal graph* is a graph with no induced cycles of length at least four. An *interval graph* is an intersection graph of line segments (intervals) on the real line, that is, its vertex set is a set of intervals, and two

vertices are adjacent if and only if their corresponding intervals intersect. A *unit-interval* graph is an intersection graph of intervals of unit length on the real line. A *clique* K in G is an (induced) subgraph, such that for any two distinct vertices $u, v \in V(K)$ we have $uv \in E(G)$. A vertex set $S \subseteq V(G)$ is a *clique* in G if $G[S]$ is a clique. A *connected component* of an undirected graph is a (vertex) maximal induced subgraph in which every two vertices are connected by a path. If a graph has only one connected component then it is called a *connected graph*. A graph is a *cluster* graph if each of its connected components are cliques. For $k \in \mathbb{N}$, a *k-cluster* graph is cluster graph with exactly k connected components. A *complete graph* is an undirected simple graph in which every pair of vertices is connected by an edge. A graph G is a *complete bipartite* graph if its vertex set can be partitioned into two disjoint (independent) sets X and Y, such that $E(G) = \{xy \mid x \in X, y \in Y\}$. For $x, y \in \mathbb{N}$, by K_{xy} we denote the complete bipartite graph on $x + y$ vertices which admits a vertex bipartition into sets X and Y of sizes x and y, respectively, such that $E(K_{xy}) = \{xy \mid x \in X, y \in Y\}$. A *tournament* is a directed graph obtained by assigning a direction to each edge of a complete graph.

2.2 Parameterized Complexity

We define common notion in Parameterized Complexity here. For the notations that are not explicitly stated here, we refer to the books [29, 53].

NP-hard problems are considered computationally hard problems for which we do not expect to get algorithms faster than exponential in the input. The paradigm of Parameterized Complexity is a way to cope with NP-hard problems. In Parameterized Complexity each problem is accompanied by a parameter together with the classical input of the problem. The key idea is to select a parameter (or combination of parameters) which is typically small on input instances in some application and find efficient algorithms where the combinatorial explosion is restricted to the parameter(s). A parameterized problem is

formally defined as follows.

Definition 1 (Definition 1.1, [29])**.** A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is finite alphabet. For an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

The size of an instance (x,k) of a parameterized problem is defined as $|x|+k$.

Fixed Parameter Tractability: A central notion in this field is of the *fixed-parameter tractability (FPT)*. It is formally defined as follows.

Definition 2 (Definition 1.2,[29])**.** A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed parameterized tractable* (FPT) if there exists an algorithm $\mathscr{A}$ (called a fixed parameter algorithm), a (non decreasing) computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x,k) \in \Sigma^* \times \mathbb{N}$ as input, the algorithm $\mathscr{A}$ correctly outputs whether $(x,k) \in L$. The algorithm $\mathscr{A}$ runs in time bounded by $f(k) \cdot |(x,k)|^c$.

The complexity class containing all fixed-parameter tractable problems is called FPT. Analogous to the notion of NP-hardness and polynomial time reductions in classical complexity, there exists a notion of hardness and reductions in the framework of Parameterized Complexity. Let us first recall the notion of polynomial time reductions that is used in the proofs of NP-hardness. A polynomial time many-one reduction from a problem $A \subseteq \Sigma^*$ to a problem $A \subseteq \Sigma^*$ is an algorithm that, given an instance x of A, runs in polynomial and outputs an instance x' of B such that x is a yes-instance of A if and only if x' is a yes-instance of B. If there exists such a reduction from a problem A to a problem B, then if B is polynomial time solvable then A is also polynomial time solvable. As the polynomial time many one reduction algorithm can be composed with the algorithm of B and a polynomial time algorithm for A can be obtained. In the following, we define an analogous notion for parameterized problems to transfer fixed parameterized tractability,

Definition 3 (Definition 13.1, [29])**.** Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from A to B is an algorithm that, given an instance (x,k) of A, outputs an instance (x',k') of B such that:

- (x,k) is a yes-instance of A if and only if (x',k') is a yes-instance of B)
- $k' \leq g(k)$ for some (non-decreasing) computable function g, and
- the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some (non-decreasing) computable function f.

If there is a parameterized reduction from A to B, and B is FPT, then A is also FPT (Theorem 13.2, [29]).

W-hardness: To classify parameterized problems according to their hardness, Downey and Fellows [44] introduced the notion of W-hierarchy. It is not required to introduce formal definitions of these classes. For our purposes in the thesis it is sufficient to say that we have the following hierarchy: $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots$. Similar to the NP-hardness, if there is a parameterized reduction from a W[t]-hard problem A to a parameterized problem B, then B is W[t]-hard. Under the assumption that $\mathsf{FPT} \neq \mathsf{W}[1]$, we can conclude that a W[1]-hard problem is not FPT. We refer the reader to [29, Chapter 13] for more details on W-hardness theory.

Kernelization: Other commonly used notion in paramterized complexity is kernelization. It is a polynomial time data reduction algorithm. Two instances of a parameterized problem L are called *equivalent* if $(x,k) \in L$ if and only if $(x',k') \in L$. A *Reduction Rule* for a parameterized problem L is a polynomial time algorithm which takes an instance (x,k) of L and returns an instance (x',k') of L. If (x,k) and (x',k') are equivalent then we say that reduction rule is *safe* or *correct*. The notion of kernelization (kernel) is defined formally as follows.

Definition 4 (Definition 2.1, [29])**.** Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. A *kernelization algorithm*, or simply *kernel* for L is an algorithm that given an instance (x,k) as an input, runs in time polynomial in $|x|+k$ and returns another instance (x',k') such that:

- $|x'| + k' \leq g(k)$ for some computable function $g \rightarrow \mathbb{N} \times \mathbb{N}$, and
- The instances (x,k) and (x',k') are equivalent.

If the upper bound $g(\cdot)$ is a polynomial function of the parameter, then we say that L admits a polynomial kernel.

If we have a kernelization algorithm for a problem, then clearly the problem is FPT. However, the converse is also true.

Proposition 1. [Lemma 2.2, [29]] *A parameterized problem L is FPT if and only if it admits a kernelization algorithm.*

Kernel lower bound: Every problem in FPT need not admit a polynomial kernel. Next, we define the notion of *polynomial compression*, which is a generalization of kernels and is useful to show non existence of polynomial kernels.

Definition 5 (Definition 15.8,[29])**.** A *polynomial compression* of a parameterized language $Q \subseteq \Sigma^* \times \mathbb{N}$ into a language $R \subseteq \Sigma^*$ is an algorithm that takes as input an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, runs in time polynomial in $|x| + k$, and returns a string y such that:

- $|y| \leq p(k)$ for some polynomial $p(\cdot)$, and
- $y \in R$ if and only if $(x,k) \in Q$.

If $|\Sigma| = 2$, the polynomial $p(\cdot)$ is called the *bit-size* of the compression.

Observe that a polynomial kernel is also a polynomial compression by treating the output kernel as an unparamterized version of A. To show kernel lower bounds, there exists a notion of *polynomial parameter transformation*. It is formally defined as follows.

Definition 6 (Definition 15.14,[29])**.** Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. An algorithm $\mathscr{A}$ is called a *polynomial parameter transformation* from A to B if given an

instance (x,k) of A, $\mathscr{A}$ works in polynomial time and outputs an equivalent instance (x',k') of B, that is, $(x,k) \in A$ if and only if $(x',k') \in B$, such that $|k'| \leq p(k)$ for some polynomial $p(\cdot)$.

The following theorem formalizes the notation of transfer of hardness.

Theorem 2.2.1 (Theorem 15.15,[29]). *Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems and assume that there exits a polynomial parameter transformation from A to B. Then, if A does not admit a polynomial compression, neither does B.*

2.3 Tools and Techniques

In this section we give some commonly used tools and techniques in the thesis.

2.3.1 k-Independence Covering Family

Definition 7 (k-independence covering family). For a graph G and an integer k, a *k-independence covering family*, denoted by $\mathscr{F}(G,k)$, is a family of independent sets of the graph G such that for any independent set X in G of size at most k there exists a set Y in $\mathscr{F}(G,k)$ such that $X \subseteq Y$.

In the following, we give algorithms to construct a k-independence covering family for a d-degenerate graph and nowhere dense graph.

d-degenerate graph: 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 [87].

The following propositions give algorithms to construct a k-independence covering family for a d-degenerate graph.

Proposition 2. [Lemma 1.1,[81]] *There exists a linear time randomized algorithm, that given as input a d-degenerate graph $H^\star$ and $k \in \mathbb{N}$, outputs an independent set Y, such that for every independent set X in $H^\star$ of size at most k the probability that X is a subset of Y is at least $(\binom{k(d+1)}{k}k(d+1))^{-1}$.*

Proposition 3. [Lemmas 3.2 and 3.3, [81]] *There are two deterministic algorithms, that given a d-degenerate graph $H^\star$ and $k \in \mathbb{N}$, outputs independence covering families $\mathscr{F}_1(H^\star,k)$ of size at most $\binom{k(d+1)}{k}2^{o(k(d+1))}\log n$ and $\mathscr{F}_2(H^\star,k)$ of size at most $\binom{k^2(d+1)^2}{k}(k(d+1))^{\mathscr{O}(1)}\log n$, respectively. These algorithms run in time $\mathscr{O}(|\mathscr{F}_1(H^\star,k)|(n+m))$ and $\mathscr{O}(|\mathscr{F}_2(H^\star,k)|(n+m))$, respectively.*

Nowhere dense graph: Towards defining the class of nowhere dense graphs, firstly we need to define the following.

Definition 8 (Shallow minor)**.** A graph M is an *r-shallow minor* of a graph G, where r is an integer, if there exists a set of disjoint subsets $V_1,\ldots,V_{|V(M)|}$ of $V(G)$ such that

1. each graph $G[V_i]$ is connected and has radius at most r, and
2. there is a bijection $\psi\colon V(M) \to \{V_1,\ldots,V_{|V(M)|}\}$ such that for every edge $uv \in E(M)$ there is an edge in G with one endpoint in $\psi(u)$ and other endpoint in $\psi(v)$.

The set of all r-shallow minors of a graph G is denoted by $G \triangledown r$. Similarly, the set of all r-shallow minors of all the members of a graph class $\mathscr{G}$ is denoted by $\mathscr{G} \triangledown r = \bigcup_{G\in\mathscr{G}}(G \triangledown r)$.

Next, we introduce the definition of a nowhere dense graph class; let $\omega(G)$ denote the size of the largest clique in G and $\omega(\mathscr{G}) = \sup_{G\in\mathscr{G}}\omega(G)$.

Definition 9 (Nowhere dense)**.** A graph class $\mathscr{G}$ is *nowhere dense* if there exists a function $f_\omega\colon \mathbb{N} \to \mathbb{N}$ such that for all r we have that $\omega(\mathscr{G} \triangledown r) \leq f_\omega(r)$.

We will mostly rely on the low treedepth colorings of nowhere dense graph classes. Towards that, we first define the notion of treedepth.

Definition 10 (Treedepth of a graph)**.** A *treedepth decomposition* of a graph G is a rooted forest F on the vertex set $V(G)$, that is, $V(F) = V(G)$, such that for every edge $uv \in E(G)$, the endpoints u and v are in ancestor-descendant relation. The *height* of the rooted forest F, denoted by $\mathbf{height}(F)$, is the maximum number of vertices on a simple path from the root of F to a leaf in F. The *treedepth* of G, denoted $\mathbf{td}(G)$, is the least $d \in \mathbb{N}$ such that there exists a treedepth decomposition F of G with $\mathbf{height}(F) = d$.

Next, we define the notion of *treedepth colorings* and state a result that shows that nowhere dense graph classes admit low treedepth colorings.

Definition 11 (Treedepth coloring)**.** An r-treedepth colouring of G is a colouring such that any $r' \leq r$ color classes induce a subgraph with treedepth at most $r'+1$. The minimum number of colors of such a coloring of G is denoted by $\mathbf{tdcolr}_r(G)$.

Proposition 4. [Theorem 5.6,[94]] *Let $\mathscr{G}$ be a nowhere dense graph class. Then there is a function $f(r,\varepsilon)$ such that $\mathbf{tdcolr}_r(G) \leq f(r,\varepsilon)\cdot n^\varepsilon$ for every integer $r \geq 0$, $G \in \mathscr{G}$, and real $\varepsilon > 0$. Furthermore, a $f(r,\varepsilon)\cdot n^\varepsilon$-treedepth colouring of G can be obtained in time $\mathscr{O}(f(r,\varepsilon)\cdot n^{1+o(1)})$.*

We refer the reader to the book by Nesetril and Ossona de Mendez [95] for a detailed exposition on nowhere dense classes of graphs, their alternate characterization and several interesting properties.

The following proposition give algorithms to construct a k independence covering family for a nowhere dense graph.

Proposition 5. [Lemmas 3.2 and 3.3, [81]]

Let H be a graph such that $H \in \mathscr{N}$, where $\mathscr{N}$ is a class of nowhere dense graphs. For any $k \in \mathbb{N}$, there are two deterministic algorithms that run in time

$$\mathscr{O}\left(f(k,\frac{1}{k})\cdot n^{1+o(1)} + g(k)\binom{k(1+d)}{k}2^{o(k(1+d))}n(n+m)\log n\right)$$

and

$$\mathscr{O}\left(f(k,\frac{1}{k})n^{1+o(1)} + g(k)\binom{k^2(1+d)^2}{k}(k(1+d))^{\mathscr{O}(1)}n(n+m)\log n\right),$$

and output a k-independence covering family for (H,k) of size $\mathscr{O}(g(k)\binom{k(1+d)}{k}2^{o(k(1+d))}n\log n)$ and $\mathscr{O}(g(k)\binom{k^2(1+d)^2}{k}(k(1+d))^{\mathscr{O}(1)}n\log n)$, respectively, where f is a function defined in Proposition 4 and $g(k) = (f(k,\frac{1}{k}))^k$.

2.3.2 Branching

Branching (or Bounded search tree) is one of the most commonly used technique in Parameterized Complexity which use the idea of backtracking. A typical branching scheme works as follows. In each branch step, algorithm identifies a small (constant size or bounded by a function of parameter) set S of eligible candidates of solution. Then the algorithm builds a partial solutions by selecting elements from S and branching into $|S|$ many subproblems with respect to each element in S. In each such step, the algorithm investigates some bounded possibilities of decisions and branches into subproblems. It can also be viewed as a search tree, which the algorithm traverses. In order to show the correctness of the algorithm it is enough to show that if the given input is a yes-instance, then a feasible solution to the problem can be discovered at one of the leaves. If the size of the search tree is bounded by a function of parameter and each branching step takes time polynomial in the input size, then the given problem is FPT.

More precisely, a branching algorithm works as follows. Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parame-

terized problem and (I,k) be an instance of L. Firstly we assign an appropriate measure $\mu(I,k)$ with (I,k), which should be bounded by a function of k. In each step the algorithm takes polynomial time and branches into *simpler* instances, say $(I_1,k_1),\ldots,(I_\ell,k_\ell)$ of L such that $(\ell \geq 2)$, ℓ is bounded by $\mu(I,k)$, (I,k) is a yes-instance of L if and only if one of the instance $(I_1,k_1),\ldots,(I_\ell,k_\ell)$ is a yes-instance of L, and each branch measure strictly decreases, that is, for all $j \in \ell$ $\mu(I_j,k_j) \leq \mu(I,k) - c$, for some constant $c > 0$. The algorithm recursively apply branching steps to instances $(I_1,k_1),\ldots,(I_\ell,k_\ell)$ and stops when each instance can be *trivially* solved. The correctness follows from the fact that if the input instance is a yes-instance then one of the branch generates a yes-instance. Since $\mu(I,k)$ is bounded by a function of k and in each branch μ strictly decreases which bounds the size of the search tree by a function of k and hence the algorithm runs in FPT time. For more details on branching technique we refer the reader to Chapter 3 of [29].

2.3.3 Iterative Compression

The iterative compression technique was invented by Reed, Smith and Vetta in 2004 [102]. The method was invented in order to design an FPT algorithm for the ODD CYCLE TRANSVERSAL problem. Later it became a commonly used technique in Parameterized Complexity to design FPT algorithms. The center of the iterative compression method is the *compression routine*. A compression routine is an algorithm, which takes an instance together with a solution of the parameterized problem as an input and either returns a solution of smaller size or return that there does not exist a solution of smaller size than the given solution. The technique iteratively builds up the input structure and find a solution at each iteration in FPT time. Then compresses the current solution using the compression routine. Generally the main goal is to use the given solution which carries some important information about the structure of the instance and design an FPT algorithm for compression routine. For more details on iterative compression technique we refer the reader to Chapter 4 of [29].

Part II

Conflict-Free Problems

Chapter 3

Conflict-free Version of Covering Problems on Graphs

Graph-modification by either deleting vertices, or deleting edges, or adding edges such that the resulting graph satisfies certain properties or becomes a member of some well-understood graph class is one of the basic problems in Graph Theory and computer science. However, most of these problems are NP-complete [111, 76], and therefore they have been extensively studied in various algorithmic paradigms that are meant to cope with NP-completeness [54, 84, 51], such as restricted classes of inputs, approximation algorithms and Parameterized Complexity. In this chapter we introduce the *conflict-free* variant of these classical problems and study them from the viewpoint of classical and Parameterized Complexity.

In the past, the conflict-free version of some classical problems have been studied, e.g. for SHORTEST PATH [66], MAXIMUM FLOW [98, 99], KNAPSACK [100], BIN PACKING [45], SCHEDULING [46], MAXIMUM MATCHING and MINIMUM WEIGHT SPANNING TREE [36, 35]. 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 [5, 6, 7]). Motivated by these works, we initiate the study of the conflict-free versions of several well studied vertex deletion problems in Parameterized Complexity. This is our main conceptual contribution in this chapter.

In the following we recall the definitions of a parameterized vertex deletion problem on graphs and their conflict-free variant. Let Π be a family of graphs (or property) – such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs or planar graphs. The vertex deletion problem corresponding to Π is formally stated as follows.

Π-VERTEX DELETION	**Parameter:** k
Input: An undirected graph G and a non-negative integer k.	
Question: Does there exist $S \subseteq V(G)$, such that $\|S\| \leq k$ and $G - S$ is in Π?	

That is, given a graph G, can we delete at most k vertices such that the resulting graph belongs to Π? The set S ***is called a*** Π***-deletion set***. The study of parameterized graph deletion problems together with their various restrictions and generalizations have been an active area of research recently.

To formulate the conflict-free version of these classical problems, let us begin with an example. Consider SET COVER, that has the following conflict-free version. We are given a universe $\mathscr{U}$ and a family $\mathscr{S}$ of subsets of $\mathscr{U}$, a positive integer k and a graph H (with $V(H) = \mathscr{S}$). The objective is to check whether there exists a $\mathscr{S}' \subseteq \mathscr{S}$ of size at most k whose union is $\mathscr{U}$ and $H[\mathscr{S}']$ is edgeless. Now, we may similarly combine the classical vertex deletion problems on graphs, with the conflict-free model described in [5, 6, 7] and arrive at the following generic conflict-free problem. Let Π be a family of graphs. The conflict-free vertex deletion problem corresponding to Π is formally stated as follows.

CONFLICT-FREE Π-VERTEX DELETION (CF-Π-VD) **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 a set $S \subseteq V(G)$, such that $|S| \leq k$, $G - S$ is in Π and S is an independent set in H?

We define CF-Π-VD, when both the graphs G, H are directed graphs (hypergraphs), appropriately. Where, the notion of independent set in directed graph is similar to that of undirected graphs. In hypergraphs, a set S of vertices in a hypergraph G is called independent if no two vertices in S are contained in an edge in G. In this chapter, we focus on CF-Π-VD problems corresponding to several well studied problems in Parameterized Complexity, namely VERTEX COVER, d-HITTING SET, SPLIT VERTEX DELETION, FEEDBACK VERTEX SET IN TOURNAMENTS (FVST) and FEEDBACK VERTEX SET (FVS). Observe that when H is the edgeless graph, CF-Π-VD is the same as Π-VERTEX DELETION and thus it generalizes the non-conflict-free version of the problem. Furthermore, when H is the same as G it corresponds to *independent* version of these problems which are also well studied, such as INDEPENDENT FEEDBACK VERTEX SET [90, 81]. Thus, CF-Π-VD is a generalization of well studied problems in algorithms and complexity.

Apart from introducing an interesting family of problems, we obtain the following results in the realm of parameterized and classical complexity. We note that several of these results are in sharp contrast to the non-conflict version of the same problem.

A *graph property* Π is a set of graphs, and a graph in Π is called a Π-*graph*. We say that Π is *hereditary* if for any graph G in Π, every induced subgraph of G is also in Π. A graph property Π has a forbidden set characterization if there is a set $\mathscr{F}$ of graphs such that a graph is a Π-graph if and only if it does not contain any graph in $\mathscr{F}$ as an induced subgraph, and further, it has a finite forbidden characterization if $\mathscr{F}$ is a finite set. We study the complexity of CF-Π-VD based on the forbidden set of the property Π.

Graph properties with finite forbidden characterization. The starting point of our results is a generic result by Cai [19] about graph properties which have a finite forbidden characterization. We show an analogous result for CF-Π-VD. In particular we show that CF-Π-VD is FPT whenever Π has a finite forbidden set characterization. Indeed, we show that this problem admits an algorithm with running time $\mathcal{O}(\alpha^k \cdot n \cdot T(m,n))$, where $T(m,n)$ is time to recognize a graph in Π and α is the size of largest graph in the finite forbidden set $\mathcal{F}$. Furthermore, it also admits a kernel with $\mathcal{O}(\alpha^2 \alpha! k^\alpha)$ vertices.

Next, we study the conflict-free version of several well-studied cases of Π-VERTEX DELETION, where Π is characterized by the finite family of forbidden induced subgraphs. For a problem $\mathcal{Q}$, we call its conflict-free version as CONFLICT-FREE Q (CF-$\mathcal{Q}$, in short). These results improve upon the generic result stated above.

1. CONFLICT FREE VERTEX COVER (CF-VC) admits a $2k$-vertex kernel, a factor 2-approximation algorithm, an $\mathcal{O}^\star(1.2738^k)$ FPT algorithm[1] and an $\mathcal{O}^\star(1.1996^n)$ exact algorithm. Furthermore, CF-VC is NP-complete even when graph G is of degree at most 2. This holds even when G is disjoint union of P_3 (P_ℓ denotes the path on ℓ vertices). Furthermore, CF-VC is polynomial time solvable when G has degree at most one, or when both G and H have a perfect matching.

2. The CONFLICT FREE d-HITTING SET (d-CF-HS) problem can be solved in $\mathcal{O}^\star(((d-1)+0.2738)^k) = \mathcal{O}^\star((d-0.7262)^k)$ time.

3. CONFLICT FREE SPLIT VERTEX DELETION (CF-SVD) can be solved in $\mathcal{O}^\star(1.2738^k k^{\mathcal{O}(\log k)})$ time and polynomial space.

4. CONFLICT FREE FEEDBACK VERTEX SET IN TOURNAMENTS (CF-FVST) can be solved in $\mathcal{O}^\star(2^k)$ time.

Let us note that given the graphs G,H, we can test whether there exists a conflict-free vertex cover (of any size) of the pair (G,H) in polynomial time by a reduction to the

[1] $\mathcal{O}^\star$ suppresses the polynomial factor in the running time.

2-SAT problem (Lemma 11). However, one can show that testing whether there exists a conflict-free feedback vertex set (of any size) is NP-complete.

Graph properties without finite forbidden characterization. Next, we consider those graph properties that are not characterized by a finite family of forbidden induced subgraphs. We show that if Π is characterized by a "well-behaved" infinite family of forbidden induced subgraphs, then CF-Π-VD is W[1]-hard. In particular, we show that CONFLICT FREE FEEDBACK VERTEX SET (CF-FVS) is W[1]-hard even when G is disjoint union of cycles. A similar result holds for CONFLICT FREE ODD CYCLE TRANSVERSAL (CF-OCT), CONFLICT FREE CHORDAL VERTEX DELETION (CF-CVD) and CONFLICT FREE INTERVAL VERTEX DELETION (CF-IVD).

This motivates us to consider some special cases of $(\mathscr{G},\mathscr{H})$-$\Pi$-VD. Here, $(\mathscr{G},\mathscr{H})$ denotes that the graph G belongs to the graph class $\mathscr{G}$ and the graph H belongs to the graph class $\mathscr{H}$. We will use $\star$ to denote that the graph is arbitrary. We show that, $(\star,\mathscr{H})$-CF-FVS ($\mathscr{H}$-CF-FVS), $\mathscr{H}$-CF-OCT, $\mathscr{H}$-CF-CVD, $\mathscr{H}$-CF-IVD are FPT, when H belongs to the family of d-degenerate graphs ($\mathscr{D}_d$), or nowhere dense graphs ($\mathscr{N}$). It is worth noting that the families of d-degenerate graphs and nowhere dense graphs include trees, graphs of bounded degree, planar graphs, graphs that exclude a fixed graph H as a minor (or a topological minor) and graphs of bounded expansion. These algorithms are based on the notion of "k-independence covering family" introduced in [81].

3.1 Preliminaries

Throughout this chapter, we use the following notions. For a (directed) graph G, we use $V(G)$ to denote the vertex set and $E(G)$ to denote the (arc) edge set of the (directed) graph G. A *conflict graph* of a graph G is a graph H, where the vertex sets of G and H are same but edge sets may be different. For two sets X and Y, by $X \setminus Y$ we mean the set $\{v | v \in X, v \notin Y\}$. For a graph G and a set $X \subseteq V(G)$, by $G[X]$ we mean the graph G

induced on X and by $G-X$ we mean the graph $G[V(G)\setminus X]$. Let E' be a subset of edges of the graph G, by $G[E']$ we mean the graph with vertex set $V(G)$ and edge set E'. For a graph G and a vertex $v \in V(G)$, by $N_G(v)$ we denote the neighborhood of v in G, that is, the set of vertices which have at least one edge incident on v in G. By $N_G[v]$ we denote the set $N_G(v) \cup \{v\}$. The subscript from the notations are ommited when it is clear from the context. Throughout the chapter given a graph G, *n and m denote the number of vertices and the edges of G, respectively*.

A graph is chordal if every cycle of length greater than 3 has a chord (an edge connecting two vertices of cycle but not part of the cycle). A complete graph is an undirected simple graph in which every pair of vertices is connected by an edge. A tournament is a directed graph obtained by assigning a direction to each edge of a complete graph.

3.2 Conflict-free Version of Properties with Forbidden Set Characterizations

3.2.1 Properties with Finite Forbidden Set Characterizations

In this subsection, we study the CF-FINITE Π-VD problem when Π is hereditary and admits a *finite forbidden set characterization*.

FPT Algorithm for CF-FINITE Π-VD**.** Let $\mathscr{F}$ be the finite forbidden set corresponding to the property Π. Cai [19] showed that the FINITE Π-VD is FPT. That is, given a graph G, testing whether there exists a set $S \subseteq V(G)$ of size at most k such that $G-S$ is a Π-graph is FPT. The algorithm works as follows. It starts by finding a forbidden vertex set X in G; among which we know that at least one vertex must go in the solution set S. Therefore, we branch on this collection of vertices, and for each vertex $v \in X$, we recursively apply the algorithm to solve the instance $(G-v, k-1)$. If one of these branches returns a Π-*deletion*

set S, then clearly $S \cup \{v\}$ is of size at most k and it is a Π-*deletion set* in G. Else, we return that the given instance is a no-instance. At every recursive call we decrease the parameter by 1, and thus, the height of the search tree does not exceed k. At every step, we branch in at most α subproblems; where α is the size of largest graph in $\mathscr{F}$. Hence, the number of nodes in the search tree does not exceed α^k. Observe that, the algorithm actually enumerates all the *minimal* Π-deletion sets of size at most k. Thus, for CF-FINITE Π-VD, all we need to do in addition, is to check whether $H[S]$ is edgeless or not. We will also need the following result for the above algorithm.

Proposition 6. [Theorem 1,[19]] *For any hereditary property* Π*, if* Π *is recognizable in time* $T(m,n)$*, then for any graph G that is not a* Π*-graph, a minimal forbidden induced subgraph of* Π *in G can be found in* $\mathscr{O}(n \cdot T(m,n))$ *time.*

With the above theorem in hand, we obtain the following theorem.

Theorem 3.2.1. CF-FINITE Π-VD *is* FPT *and admits an algorithm with running time* $\mathscr{O}(\alpha^k \cdot n \cdot T(m,n))$*, where* $T(m,n)$ *is the time to recognize a graph in* Π *and* α *is the size of largest graph in the finite forbidden set* $\mathscr{F}$*.*

3.2.2 A Polynomial Kernel for CF-FINITE Π-VD

In this section, we design a polynomial kernel for CF-FINITE Π-VD. Towards that, we define following reduction rules. The first reduction rule is based on a simple observation that the vertices that do not participate in any forbidden induced subgraph of G can be deleted.

Reduction Rule 1. *Let* (G,H,k) *be an instance of* CF-FINITE Π-VD*. If G contains a vertex v, which is not part of any forbidden induced subgraph in G, delete v from G and H. The new instance is* $(G-v,H-v,k)$.

Safeness of Reduction Rule 1 follows from the fact that G and $G-v$ have the same family of forbidden sets that needs to be hit. Also, we can test whether a vertex v is part

of any forbidden induced subgraph or not by enumerating all subsets X of size $\alpha - 1$ and testing whether $G[X \cup \{v\}]$ is a forbidden set or not. This implies that we can apply this reduction rule in polynomial time. Next reduction rule is based on the classical Sunflower Lemma. A *sunflower* with k petals and a core Y is collection of k disjoint sets $S_1, \cdots, S_k$ such that $S_i \neq S_j$ for all $i \neq j$, and they intersect exactly at Y. The sets $S_i \setminus Y$ are called *petals* and are non empty. The set Y is called the core.

Lemma 1. [29, Theorem 2.25] *Let $\mathscr{A}$ be the family of sets of size exactly d over a universe $\mathscr{U}$. If $|\mathscr{A}| > d!(k-1)^d$, then $\mathscr{A}$ contains a sunflower with k petals, and such a sunflower can be computed in time polynomial in $|\mathscr{A}|$, $|\mathscr{U}|$ and k.*

Given a graph G and a set of finite forbidden graphs $\mathscr{F} = \{F_1, \ldots, F_q\}$, we first enumerate all forbidden induced subgraphs of G in polynomial time. Let $\mathscr{A}$ be the family of vertex sets of induced forbidden subgraphs of G and η denotes the size of maximum sized set in $\mathscr{A}$. We start with the following simple but useful observation.

Observation 3.2.2. *(G, H, k) is a* yes*-instance of* CF-FINITE Π-VD *if and only if there exists a vertex subset $S \subseteq V(G)$ of size at most k such that S intersects all the sets in $\mathscr{A}$ and $H[S]$ is edgeless.*

For a family $\mathscr{A}^*$ of subsets of sets of $V(G)$, by $U(\mathscr{A}^*)$, we denote the union of set of vertices in all sets in $\mathscr{A}^*$.

Observation 3.2.3. *Suppose that, $\mathscr{A}$ contains a sunflower $S_1, \ldots, S_{k+2}$ with $k+2$ petals and a core Y. Let $\mathscr{A}' = \mathscr{A} \setminus \{S_{k+2}\}$ and $H' = H[U(\mathscr{A}')]$. Then, there exists a vertex subset $S \subseteq U(\mathscr{A})$ of size at most k such that S intersects all the sets in $\mathscr{A}$ and $H[S]$ is edgeless if and only if there exists a vertex subset $S' \subseteq V(\mathscr{A}')$ of size at most k such that S' intersects all the sets in $\mathscr{A}'$ and $H'[S']$ is edgeless.*

Proof. In the forward direction, assume that there exists a vertex subset $S \subseteq U(\mathscr{A})$ of size at most k such that S intersects all the sets in $\mathscr{A}$ and S is an independent set in H. Let

$S' = S \cap U(\mathscr{A}')$. The only vertices that appears in S but do not appear in S' are those which are present in S_{k+2} but do not *appear* in any other set of $\mathscr{A}$. This implies that S' intersects all the sets in $\mathscr{A}'$. Furthermore, since $S' \subseteq S$, we have that $H'[S']$ is edgeless.

In the reverse direction, assume that there exists a vertex subset $S' \subseteq U(\mathscr{A}')$ of size at most k such that S' intersects all the sets in $\mathscr{A}'$ and S' is an independent set in H'. We will show that S' itself intersects all the sets in $\mathscr{A}$. Suppose not then S' does not hit $S_{k+2} \in \mathscr{A}$. Since $\mathscr{A}'$ contains sunflower $S_1, \ldots, S_{k+1}$, S' must contain a vertex from Y, otherwise S' contains a vertex from each $S_i \setminus Y$, $i \in [k+1]$ which is a contradiction to the size of S'. However, a vertex from Y hits S_{k+2} in $\mathscr{A}$. Hence, S' intersects all the sets in $\mathscr{A}$. Observe that, since H' is an induced subgraph of H, we have that $H[S']$ is edgeless. This completes the proof. □

Next, we apply the following procedure to reduce the family $\mathscr{A}$. If $\mathscr{A}$ contains a sunflower $S_1, \ldots, S_{k+2}$ with $k+2$ petals and a core Y, then delete S_{k+2} from $\mathscr{A}$ and repeat on $\mathscr{A} = \mathscr{A} \setminus S_{k+2}$. Otherwise, return $\mathscr{A}$. Let $\mathscr{A}'$ be the resultant family. Using Lemma 1, the above procedure can be applied in polynomial time.

Reduction Rule 2. *Let v be a vertex in G such that v is not in $V(\mathscr{A}')$, then delete v from G and H. The new instance is $(G-v, H-v, k)$.*

Lemma 2. *Reduction Rule 2 is safe.*

Proof. The proof follows from Observations 3.2.2,3.2.3 and the construction of the family $\mathscr{A}'$. □

Theorem 3.2.4. CF-FINITE Π-VD *admits a kernel with at most $\mathcal{O}(\eta^2 \eta! k^\eta)$ vertices, where η is the size of largest graph in the finite forbidden set $\mathscr{F}$ corresponding to* Π.

Proof. Given an instance (G, H, k) of CF-FINITE Π-VD, the kernelization algorithm works as follows. Firstly, the algorithm applies Reduction Rule 1 exhaustively. Next, the algorithm constructs a family $\mathscr{A}$ of vertex sets of induced forbidden subgraphs of G and

applies Reduction Rule 2 exhaustively and get the reduced family $\mathscr{A}'$. Let V' denotes the set of vertices present in any forbidden set in $\mathscr{A}'$. Let $(G' = G[V'], H' = H[V'], k)$ be the reduced instance. By safeness of Reduction Rule 1, Observations 3.2.2,3.2.3, and Lemma 2 it follows that, (G,H,k) is a yes-instance of CF-FINITE Π-VD if and only if (G',H',k) is a yes-instance of CF-FINITE Π-VD. Since $V(G')$ is same as the set of vertices present in any set in $\mathscr{A}'$, it is sufficient to bound size of family $\mathscr{A}'$ to bound the size of the graph G'. If for some $\eta' \in \{1,\ldots,\eta\}$, the number of sets in $\mathscr{A}'$ of size exactly η' is more than $\eta'!(k+1)^{\eta'}$, then by Lemma 1, we can find a sunflower with $k+2$ petals and apply Reduction Rule 2. Since, each set can be of size at most η in $\mathscr{A}'$, therefore $|V(G')| \leq \eta^2\eta!(k+1)^{\eta}$. This completes the proof of theorem. □

3.2.3 Properties that do not admit finite forbidden characterization

It is well know that a property Π is hereditary if and only if Π admits a forbidden set characterization [19]. Let $\mathscr{F}$ denote the forbidden set corresponding to Π. Following the previous section, a natural question that arises is what happens when $\mathscr{F}$ is *infinite*. We call the corresponding vertex deletion problem as CONFLICT FREE INFINITE Π-VERTEX DELETION (CF-INFINITE Π-VD). For example, suppose that Π is a family of forests, or chordal graphs, or interval graphs, or bipartite graphs. Then, the corresponding classical problems of Π-VERTEX DELETION (Π-VD) problems are known as FEEDBACK VERTEX SET (FVS), CHORDAL VERTEX DELETION (CVD), INTERVAL VERTEX DELETION (IVD) and ODD CYCLE TRANSVERSAL (OCT) and these problems are known to be FPT [24, 70, 86, 102]. However, we will show now that conflict-free version of these problems is W[1]-hard. Indeed, CONFLICT FREE FEEDBACK VERTEX SET (CF-FVS) is W[1]-hard, when parameterized by solution size, even when the graph G belongs to the class of graphs of *disjoint union of cycles* ($\mathscr{C}$).

Towards this, we give a parameter preserving reduction from MULTICOLORED INDEPENDENT SET (MCIS) problem to ($\mathscr{C},\star$)-CF-FVS ($\mathscr{C}$-CF-FVS). It is well known

that the MCIS problem is W[1]-hard. See [29] for further details on the notion of W[1]-hardness and for the fact that MCIS is W[1]-hard. The MCIS problem is formally defined as follows.

MULTICOLORED INDEPENDENT SET (MCIS) **Parameter:** k
Input: A graph G, an integer k, and a partition of $V(G)$ into k sets $V_1, \ldots, V_k$.
Question: Does there exist a set $X \subseteq V(G)$, such that $X \cap V_i = 1$, $\forall i \in [k]$ and $G[X]$ is edgeless?

Given an instance $(G, (V_1, \ldots, V_k), k)$ of MCIS, we construct an instance (G', H, k) of CF-FVS as follows. We let $H = G$. Next, we construct the graph G'. Vertices in $V(G') = V(H) = V(G)$. For each set V_i, $i \in [k]$, we construct a cycle $C_{|V_i|}$ (C_l denotes cycle on l vertices) on vertex set V_i in G'. Clearly, this construction can be carried out in polynomial time, and graph $G \in \mathscr{C}$.

Lemma 3. *$(G, (V_1, \ldots, V_k), k)$ is a* yes*-instance of* MCIS *if and only if (G', H, k) is a* yes*-instance of $\mathscr{C}$-*CF-FVS.

Proof. In the forward direction, let $(G, (V_1, \ldots, V_k), k)$ is a yes-instance of MCIS and let S be its solution of size at most k. We claim that, S is also a solution to (G', H, k) of $\mathscr{C}$-CF-FVS. Observe that, $S \cap V_i = 1$ for each $i \in [k]$. By our construction, the graph G' is disjoint union of cycles $C_{|V_i|}$ on vertex set V_i, for each $i \in [k]$. This implies that, S hits all cycles in G'. On the other hand, since H is isomorphic to the graph G and $G[S]$ is edgeless, we have that $H[S]$ is also edgeless.

Conversely, let (G', H, k) is a yes-instance of $\mathscr{C}$-CF-FVS and let S' be its solution of size at most k. We claim that, S' is also a solution to $(G, (V_1, \ldots, V_k), k)$ of MCIS. By the construction of G', there exist k disjoint cycles in G' on each set V_i, $i \in [k]$. Hence, $S' \cap V_i = 1$. Also, $H[S']$ is edgeless and graph G is isomorphic to graph H, therefore $G[S']$ is also edgeless. This completes the proof. □

We obtain the following theorem using construction, Lemma 3 and W[1]-hardness of

MCIS.

Theorem 3.2.5. *$\mathscr{C}$-CF-FVS is $\mathsf{W}[1]$-hard, when parameterized by the solution size, where $\mathscr{C}$ is the family of graphs which are disjoint union of cycles.*

The proof of Theorem 3.2.5 requires nothing specific about $\mathscr{C}$-CF-FVS, except that G is a disjoint union of forbidden sets where each forbidden set is identified with a color class V_i. If $\mathscr{F}$ is infinite and *well behaved* in the following sense: given an integer n we can output a forbidden set F of size polynomial in n (in fact, size $f(k) \cdot n^{\mathscr{O}(1)}$ will also work for our purpose) in time $\tau(k) \cdot n^{\mathscr{O}(1)}$, then we can mimic the proof of Theorem 3.2.5 and show that the corresponding CF-Π-VD is $\mathsf{W}[1]$-hard. Let us note that in certain cases, e.g. for bipartite graphs where the family of forbidden subgraphs are odd cycles, we may need to augment a color class V_i with additional vertices to obtain a forbidden set in $\mathscr{F}$ in the graph G. This is easily handled by making the additional vertices adjacent to all vertices in the conflict graph H, which ensures that they cannot be selected in any solution of cardinality greater than one. In particular, this holds for Π being the family of chordal graphs, or interval graphs, or bipartite graphs. Here, f and τ are computable functions.

3.2.4 Results on properties without finite forbidden characterization

In Section 3.2.3, we have shown that if $\mathscr{F}$ is infinite, CF-Π-VD is $\mathsf{W}[1]$-hard in general, even though the corresponding classical problem is FPT, e.g. CF-FVS, CF-OCT, CF-CVD etc. In light of this, a natural question that arises is what happens if H is restricted to certain graph classes. In this section, we show that CF-INFINITE Π-VD is FPT when H is restricted to the class of *d-degenerate graphs* or *no-where dense graphs*. We refer to these restricted variants of problem as $\mathscr{D}_d$-CF-Π-VD and $\mathscr{N}$-CF-Π-VD, respectively.

The *degeneracy* of an n-vertex graph $G^\star$ is defined as the minimum integer d such that there exists an ordering $\sigma : V(G^\star) \to \{1, \cdots, n\}$ where every vertex v has at most d neighbors u for which $\sigma(u) > \sigma(v)$. Such an ordering σ is called a *d-degeneracy sequence*

of the graph $G^\star$. We fix one such sequence, and then for any vertex $v \in V(G^\star)$, we define its *forward* and *backward* neighbors in $G^\star$ with respect to this ordering. Our algorithm is based on the construction of a k-independence covering family of a graph, using the Independence Covering Lemma of [81]. We recall the notion of k-independence covering family and Independence Covering Lemma here. For a graph $H^\star$ and an integer k, a *k-independence covering family*, denoted by $\mathscr{F}(H^\star,k)$, is a family of independent sets of the graph $H^\star$ such that for any independent set X in $H^\star$ of size at most k there exists a set Y in $\mathscr{F}(H^\star,k)$ such that $X \subseteq Y$. We will use the following propositions to construct a k-independence covering family for H.

Proposition 7. [Lemma 1.1,[81]] *There exists a linear time randomized algorithm, that given as input a d-degenerate graph $H^\star$ and $k \in \mathbb{N}$, outputs an independent set Y, such that for every independent set X in $H^\star$ of size at most k the probability that X is a subset of Y is at least $(\binom{k(d+1)}{k}k(d+1))^{-1}$.*

Proposition 8. [Lemmas 3.2 and 3.3,[81]] *There are two deterministic algorithms, that given a d-degenerate graph $H^\star$ and $k \in \mathbb{N}$, outputs independence covering families $\mathscr{F}_1(H^\star,k)$ of size at most $\binom{k(d+1)}{k}2^{o(k(d+1))}\log n$ and $\mathscr{F}_2(H^\star,k)$ of size at most $\binom{k^2(d+1)^2}{k}(k(d+1))^{\mathscr{O}(1)}\log n$, respectively. These algorithms run in time $\mathscr{O}(|\mathscr{F}_1(H^\star,k)|(n+m))$ and $\mathscr{O}(|\mathscr{F}_2(H^\star,k)|(n+m))$, respectively.*

In the following, we present an FPT algorithm for $\mathscr{D}_d$-CF-Π-VD problems. The algorithm is based on the observation that, given an independence covering family of conflict graph, the conflict-free solution of the problem lies inside one of the sets in this family. By construction, each set in this family is an independent set in H, and therefore the problem of finding a solution to the given instance of $\mathscr{D}_d$-CF-Π-VD boils down to finding a solution of Π-VD in the graph G that also lies in a chosen set in the family. In particular, it reduces to solving the following annotated version of Π-VD.

ANNOTATED-Π-VD (A-Π-VD)	**Parameter:** k
Input: A graph G, $Y \subseteq V(G)$ and an integer k.	
Question: Does there exist $S \subseteq Y$ of size at most k such that $G - S$ is in Π?	

Theorem 3.2.6. *Let* Π *be a property such that* A-Π-VD *admits an algorithm with running time* $\tau(k)n^{\mathcal{O}(1)}$. *Then,* $\mathscr{D}_d$-CF-Π-VD *admits a randomized algorithm with running time* $\binom{k(d+1)}{k}k(d+1)\tau(k)n^{\mathcal{O}(1)}$, *and a deterministic algorithm with running time* $\min\left\{\binom{k(d+1)}{k}2^{o(k(d+1))}\log n,\ \binom{k^2(d+1)^2}{k}(k(d+1))^{\mathcal{O}(1)}\log n\right\}\tau(k)n^{\mathcal{O}(1)}$.

Proof. First, we give the randomized algorithm. Given an instance (G,H,k) of $\mathscr{D}_d$-CF-Π-VD we do as follows. Run the following two step procedure $(\binom{k(1+d)}{k}k(d+1))$ times.

1. Run the algorithm in Proposition 7 on (H,k), and obtain the set Y.
2. Solve A-Π-VD on the instance (G,Y,k), using the algorithm running in time $\tau(k)n^{\mathcal{O}(1)}$.

The algorithm outputs *y*es, if Step 2 returns *y*es at least once, else algorithm returns *n*o. Now we prove the correctness of algorithm. Since in Step 1, the output set Y is an independent set in the conflict graph H, if the algorithm returns yes, then the input instance is a yes-instance. Now suppose that the input instance is a yes-instance, and X be its solution. By Proposition 7, probability that $X \subseteq Y$ is at least $p = (\binom{k(d+1)}{k}k(d+1))^{-1}$. We repeat the procedure $1/p$ times, so the probability that in all executions $X \nsubseteq Y$ is at most $(1-p)^{1/p} \leq 1/e$. Therefore, algorithm returns *y*es with probability at least $1-1/e$. Running time follows from Proposition 7, and the assumed running time of the algorithm for A-Π-VD.

Next, we give the deterministic algorithm. Given an instance (G,H,k) of $\mathscr{D}_d$-CF-Π-VD, the algorithm works as follows. Algorithm first constructs k-independence covering family $\mathscr{F}(H,k)$ of the conflict graph H, using Proposition 8. Then, for all sets $Y \in \mathscr{F}(H,k)$, algorithm solves A-Π-VD on instance (G,Y,k), using the algorithm assumed in the

statement of the theorem. The algorithm outputs *yes* if for some set $Y \in \mathscr{F}(H,k)$, the A-Π-VD returns *yes*, otherwise returns *no*. The correctness of the algorithm follows from the definition of k-independence covering family. The running time follows from Proposition 8, and the assumed running time of the algorithm for A-Π-VD. This completes the proof. □

The above theorem naturally leads to the question of when can A-Π-VD be FPT. We give an affirmative answer for several cases when the integer weighted version (W-Π-VD) of the corresponding Π-VD is FPT. For a graph G, a weight function $w : V(G) \to \mathbb{N}$, and a set $S \subseteq V(G)$, we define the weight of S as $\sum_{v \in S} w(v)$. The W-Π-VD problem is defined formally as follows.

WEIGHTED Π-VD (W-Π-VD) **Parameter:** k
Input: A graph G, a weight function $w : V(G) \to \mathbb{N}$, and an integer k.
Question: Does there exist $S \subseteq V(G)$ of weight at most k, such that $G - S$ is in Π?

We give a polynomial time reduction from A-Π-VD to W-Π-VD. Towards this, given an instance (G,Y,k) of A-Π-VD, we construct an instance (G',w,k) of W-Π-VD as follows. We take the graph G' identical to the graph G. The weight function w is defined as follows. We assign $w(v) = k+1$ if $v \in V(G) \setminus Y$, otherwise $w(v) = 1$.

Lemma 4. *(G,Y,k) is a* yes*-instance of* A-Π-VD *if and only if (G',w,k) is a* yes*-instance of* W-Π-VD.

Proof. In the forward direction, let (G,Y,k) is a yes-instance of A-Π-VD, and let S be its minimal solution of size at most k. We claim that, S is also a solution to (G',w,k) of W-Π-VD. Since $S \subseteq Y$, we have that $w(v) = 1$, for every $v \in S$. Therefore, the weight of S is at most k. Since G' is isomorphic to G, $G' - S$ is also a Π-graph.

Conversely, let (G',w,k) is a yes-instance of W-Π-VD, and let S' be its solution of weight at most k. We claim that, S' is also a solution to (G,Y,k) for A-Π-VD. Observe

that, all the vertices in $V(G') \setminus Y$ have weight $k+1$, therefore, $S' \subseteq Y$. Since, every vertex in S' has weight 1, $|S'| \leq k$. Furthermore, since the graph G is isomorphic to the graph G', we have that $G - S'$ is in Π. This completes the proof. □

Using construction and Lemma 4, we obtain the following lemma.

Lemma 5. *Let* Π *be a property such that* W-Π-VD *admits an algorithm with running time* $\gamma(k)n^{\mathcal{O}(1)}$. *Then,* A-$\Pi$-VD *also admits an algorithm with running time* $\gamma(k)n^{\mathcal{O}(1)}$.

It is known that WEIGHTED FEEDBACK VERTEX SET (WFVS) can be solved in time $\mathcal{O}(3.618^k n^{\mathcal{O}(1)})$ [26] and thus, by Lemma 5, we have that A-FVS can be solved in time $\mathcal{O}(3.618^k n^{\mathcal{O}(1)})$. By applying Theorem 3.2.6, we obtain the following corollary.

Corollary 1. $\mathscr{D}_d$-CF-FVS *either admits a randomized algorithm with running time* $\binom{k(d+1)}{k}k(d+1)\tau(k)n^{\mathcal{O}(1)}$, *or a deterministic algorithm with running time* $\min\left\{\binom{k(d+1)}{k}2^{o(k(d+1))}\log n, \binom{k^2(d+1)^2}{k}(k(d+1))^{\mathcal{O}(1)}\log n\right\}\tau(k)n^{\mathcal{O}(1)}$. *Here,* $\tau(k) = 3.618^k$.

3.2.5 CONFLICT FREE ODD CYCLE TRANSVERSAL

Let Π be the family of bipartite graphs. Then, the corresponding CF-Π-VD is a conflict-free version of OCT, namely, CONFLICT FREE ODD CYCLE TRANSVERSAL (CF-OCT). In the following, we design an FPT algorithm for $\mathscr{D}_d$-CF-OCT. Towards this, we use the result for ANNOTATED-OCT (A-OCT) problem. It is known that A-OCT can be solved in time $\mathcal{O}(4^k k^6(n+m))$ [81]. Thus, by applying Theorem 3.2.6, we obtain the following corollary.

Corollary 2. $\mathscr{D}_d$-CF-OCT *either admits a randomized algorithm with running time* $\binom{k(d+1)}{k}k(d+1))\tau(k)n^{\mathcal{O}(1)}$, *or a deterministic algorithm with running time* $\min\left\{\binom{k(d+1)}{k}2^{o(k(d+1))}\log n, \binom{k^2(d+1)^2}{k}(k(d+1))^{\mathcal{O}(1)}\log n\right\}\tau(k)n^{\mathcal{O}(1)}$. *Here,* $\tau(k) = 4^k k^6$.

3.2.6 CONFLICT FREE CHORDAL VERTEX DELETION

Let Π be the family of chordal graphs. Then, the corresponding CF-Π-VD is a conflict-free version of CVD, namely, CONFLICT FREE CHORDAL VERTEX DELETION (CF-CVD). In the following, we design an FPT algorithm for $\mathscr{D}_d$-CF-CVD. Towards this, we give a polynomial time parameter preserving reduction from ANNOTATED-CVD (A-CVD) to the CVD problem. Given an instance (G,Y,k) of A-CVD, in the following, we give a construction to generate an instance (G',k) of CVD.

Construction 1. To construct the graph G', we replace each vertex $v \in V(G) \setminus Y$ with a $k+1$ sized vertex set $X_v = \{v_1, \ldots, v_{k+1}\}$. The vertex set of G' is $\cup_{v \in V(G) \setminus Y} X_v \cup Y$. If uv is an edge in G such that $u, v \in Y$, then we add uv to G'. Next, for each vertex $v \in V(G) \setminus Y$, we introduce edges $v_i v_j$, for all $v_i, v_j \in X_v, i \neq j$. Furthermore, for each edge $vu \in E(G)$ such that $u \in Y$, we introduce edges $v_i u$ in G' for all $v_i \in X_v$, and when $u \in V(G) \setminus Y$, we introduce edges $v_i u_j$ in G' such that $v_i \in X_v, u_j \in X_u, 1 \leq i, j \leq k+1$. Observe that, for a vertex $v \in V(G) \setminus Y$, $G'[X_v]$ is a clique of size $k+1$ in G'. Let S_u be the set of vertices added in G' corresponding to a vertex u in G. Observe that, for an edge $uv \in E(G)$, we make a complete bipartite graph between vertices of sets S_u, S_v in G'.

Lemma 6. *(G,Y,k) is a* yes*-instance of* A-CVD *if and only if (G',k) is a* yes*-instance of* CVD.

Proof. In the forward direction, let (G,Y,k) is a yes-instance of A-CVD, and let S be its solution of size at most k. We claim that S is also a solution to (G',k) for CVD. For a contradiction, assume that there exists a chordless induced cycle C in $G' - S$ of length at least 4. Observe that, C can contain at most one vertex from a set S_v. Indeed, since C has at least 4 vertices, if two vertices, say $v_i, v_j \in S_v$ are present in C, and let uv_i is an edge in C then, by construction of G', we have that edges $v_i v_j, uv_j \in E(G')$ and it contradicts the assumption that C is an induced cycle of length at least 4. This implies that, we can now replace a vertex v_i in C by the corresponding vertex v in G and get a chordless cycle in

$G-S$. This contradicts that $G-S$ is a chordal graph.

Conversely, let (G',k) is a yes-instance of CVD and let S' be its solution of size at most k. We claim that S' is also a solution of A-CVD in (G,Y,k). Observe that, for a vertex $v \in V(G) \setminus Y$, the neighborhood of vertices in the set S_v is same in G'. Thus, if a vertex $v_i \in S_v$ participates into a chordless induced cycle C, then replacing v_i by a different vertex $v_j \in S_v$, we get another chordless induced cycle C'. Therefore, if a minimal solution S' contains a vertex $v_i \in S_v$, it must contain all the vertices in S_v, which would contradict the size of S'. Therefore, S' does not contain any vertex from S_v, $v \in V(G) \setminus Y$. Thus, $S' \subseteq Y$ and is of size at most k. Also, observe that, G is an induced subgraph of G' and chordal graphs are closed under induced subgraphs. This implies that, $G-S$ is also a chordal graph. □

Lemma 6 implies that, it is sufficient to solve CVD on (G',k) to solve A-CVD on (G,Y,k). It is well known that CVD can be solved in time $\mathcal{O}(2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)})$ [25]. Hence, we obtain the following result.

Lemma 7. A-CVD *can be solved in time* $\mathcal{O}(2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)})$.

Combining Theorem 3.2.6 and Lemma 7, we obtain the following corollary.

Corollary 3. $\mathscr{D}_d$-CF-CVD *either admits a randomized algorithm with running time* $\binom{k(d+1)}{k}k(d+1))\tau(k)n^{\mathcal{O}(1)}$, *or a deterministic algorithm with running time* $\min\left\{\binom{k(d+1)}{k}2^{o(k(d+1))}\log n, \binom{k^2(d+1)^2}{k}(k(d+1))^{\mathcal{O}(1)}\log n\right\}\tau(k)n^{\mathcal{O}(1)}$. *Here,* $\tau(k) = 2^{\mathcal{O}(k\log k)}$.

3.2.7 Conflict Free Interval Vertex Deletion

Let Π be the family of interval graphs. Then, the corresponding CF-Π-VD is a conflict-free version of IVD, namely, Conflict Free Interval Vertex Deletion (CF-IVD). In the following, we design an FPT algorithm for $\mathscr{D}_d$-CF-IVD. Towards this, we give a

polynomial time parameter preserving reduction from ANNOTATED-IVD (A-IVD) to the IVD problem.

We use the characterization that a graph is an *interval graph* if and only if it is *chordal* and *AT-free*. Three vertices form an *asteroidal triple* in a graph G, if every two of them are connected by a path avoiding the neighborhood of the third. A graph is AT-free if it has no asteroidal triple. Given an instance (G,Y,k) of A-IVD, we construct an instance (G',k) of IVD, using Construction 1 described in Section 3.2.6.

Lemma 8. *(G,Y,k) is a* yes*-instance of* A-IVD *if and only if (G',k) is a* yes*-instance of* IVD.

Proof. In the forward direction, let (G,Y,k) is a yes-instance of A-IVD, and let S be its solution of size at most k. We claim that S is also a solution to (G',k) of IVD. Observe that, $G'-S$ is chordal using arguments similar to the proof of Lemma 7. Now it is only remaining to prove that $G'-S$ is also AT-free. For a contradiction, assume that there exists an asteroidal triple $A=\{u,v,w\}$ in $G'-S$. Observe that for a vertex, say u in A, if $u \in V(G) \setminus Y$ then v,w cannot belong to the set S_u. For contradiction, assume that v belongs to S_u. Since neighbours of u and v are same, all paths from u to w passes through neighbour of v, which is contradiction to the fact that A is an asteroidal triple. Therefore, all three vertices in A correspond to different vertices in G. Observe that, a minimal path P between any two vertices, say between u and v, that avoids neighborhood of the third vertex w, cannot contain a vertex from S_w if $w \in V(G) \setminus Y$, since w is adjacent to all the vertices in S_w. If u or v belongs to $V(G) \setminus Y$, then since P is a minimal path it cannot contain any other vertices from S_u or S_v apart from u and v, since the neighborhood of all the vertices in S_u and S_v are same. Therefore, we can replace vertices in P with corresponding vertices in G, and we get an asteroidal triple corresponding to $\{u,v,w\}$ in G, which is a contradiction to the fact that S is a solution to (G,Y,k) of A-IVD.

Conversely, let (G',k) is a yes-instance of IVD, and let S' be its minimal solution of size at most k. We claim that S' is also a solution to (G,Y,k) of A-IVD. Observe

that, $G-S$ is chordal using arguments similar to proof of Lemma 7. Now it is only remaining to prove that $G-S$ is also AT-free. Observe that, to delete an asteroidal triple either we have to delete a vertex from asteroidal triple or we have to delete all paths between at least two vertices of asteroidal triple that does not passes through third vertex or neighborhood of third vertex. We claim that, S' does not contain any vertex from S_v, $v \in V(G) \setminus Y$. For contradiction, assume that there exist a vertex v_i corresponding to a vertex $v \in V(G) \setminus Y$ in S' and participates into an asteroidal triple $\{v_i, u, w\}$ then replacing v_i by a vertex $v_j \in S_v, i \neq j$ we get another asteroidal triple $\{v_j, u, w\}$. Therefore, if S' contains a vertex $v_i \in S_v$, it will contain all vertices in S_v, which would contradict that S' is of size at most k. Now consider the case when the vertex v_i does not participate in any asteroidal triple, then v_i is in S' to delete a path between vertices of an asteroidal triple but then replacing v_i by a vertex $v_j \in S_v, i \neq j$, we get another path and there will be $k+1$ such paths. Therefore, taking v_i in S' contradicts that S' is minimal. This implies that $S' \subseteq Y$ and is of size at most k. Also, observe that, G is an induced subgraph of G' and interval graphs are closed under induced subgraph. Therefore, $G-S$ is also an interval graph. This completes the proof.

□

Lemma 8 implies that, it is sufficient to solve IVD on (G', k) to solve A-IVD on (G, Y, k). It is known that IVD can be solved in time $\mathcal{O}(8^k(n+m))$ [21]. Hence, we obtain the following result.

Lemma 9. A-IVD *can be solved in time* $\mathcal{O}(8^k k^2(n+m))$.

Combining Theorem 3.2.6 and Lemma 9, we obtain the following corollary.

Corollary 4. $\mathscr{D}_d$-CF-IVD *either admits a randomized algorithm with running time* $\binom{k(d+1)}{k} k(d+1))\tau(k)n^{\mathcal{O}(1)}$, *or a deterministic algorithm with running time* $\min\left\{\binom{k(d+1)}{k}2^{o(k(d+1))}\log n, \binom{k^2(d+1)^2}{k}(k(d+1))^{\mathcal{O}(1)}\log n\right\}\tau(k)n^{\mathcal{O}(1)}$. *Here,* $\tau(k) = 8^k$.

3.2.8 Nowhere Dense Graphs

In this section, we present an FPT algorithm for CF-INFINITE Π-VD when H belongs to *nowhere dense* class of graphs ($\mathscr{N}$-CF-Π-VD). This class of graphs is a generalization of bounded tree-depth classes, bounded local tree-width classes, planar graphs, bounded genus, excluded minors, excluded topological minor and bounded local expansion classes [95]. The algorithms is again based on the fact that these classes of graphs also admit independence covering family. See Section 2.3.1 for the definition of the class of nowhere dense graphs. We recall the following result to find a k-independence covering family for nowhere dense graphs.

Proposition 9. [Lemmas 3.2 and 3.3,[81]] *Let H be a graph such that $H \in \mathscr{N}$, where $\mathscr{N}$ is a class of nowhere dense graphs. For any $k \in \mathbb{N}$, there are two deterministic algorithms that run in time*

$$\mathscr{O}\left(f(k,\frac{1}{k})\cdot n^{1+o(1)} + g(k)\binom{k(1+d)}{k}2^{o(k(1+d))}n(n+m)\log n\right)$$

and

$$\mathscr{O}\left(f(k,\frac{1}{k})n^{1+o(1)} + g(k)\binom{k^2(1+d)^2}{k}(k(1+d))^{\mathscr{O}(1)}n(n+m)\log n\right),$$

and output a k-independence covering family for (H,k) of size $\mathscr{O}(g(k)\binom{k(1+d)}{k}2^{o(k(1+d))}n\log n)$ and $\mathscr{O}(g(k)\binom{k^2(1+d)^2}{k}(k(1+d))^{\mathscr{O}(1)}n\log n)$, respectively, where f is a function defined in Proposition 4 and $g(k) = (f(k,\frac{1}{k}))^k$.

Then, analogous to Theorem 3.2.6, we can obtain Theorem 3.2.7 and it's corollaries.

Theorem 3.2.7. *Let Π be a property such that* A-Π-VD *admits an algorithm with running time $\tau(k)n^{\mathscr{O}(1)}$. Then, $\mathscr{N}$-*CF-Π-VD *admit deterministic algorithms with running time $\mathscr{O}\left(\left(f(k,\frac{1}{k}) + g(k)\binom{k(1+d)}{k}2^{o(k(1+d))}\right)\tau(k)n^{\mathscr{O}(1)}\right)$ and $\mathscr{O}\left(\left(f(k,\frac{1}{k}) + g(k)\binom{k^2(1+d)^2}{k}(k(1+d))^{\mathscr{O}(1)}\right)\tau(k)n^{\mathscr{O}(1)}\right)$. Here, f is a function defined in Proposition 4 and $g(k) = (f(k,\frac{1}{k}))^k$.*

Corollary 5. *$\mathscr{N}$-CF-FVS, $\mathscr{N}$-CF-OCT, $\mathscr{N}$-CF-CVD and $\mathscr{N}$-CF-IVD admit* FPT *algorithms.*

3.3 Well Studied Special Cases of CF-FINITE Π-VD

We can obtain improved algorithms for the conflict-free version of several well-studied cases of Π-VERTEX DELETION whenever Π is characterized by the finite family of forbidden induced subgraphs. In this section, we give improved algorithms for CONFLICT FREE VERTEX COVER, CONFLICT FREE d-HITTING SET, CONFLICT FREE SPLIT VERTEX DELETION and CONFLICT FREE FEEDBACK VERTEX SET IN TOURNAMENTS.

3.3.1 CONFLICT FREE VERTEX COVER

In this section, we study the conflict-free version of the classical VERTEX COVER, namely CONFLICT FREE VERTEX COVER (CF-VC). In particular, we study the following problem.

CONFLICT FREE VERTEX COVER (CF-VC) **Parameter:** k
Input: A graph $G = (V, E)$, a conflict graph H, and an integer k.
Question: Does there exist $X \subseteq V(G)$ of size at most k, such that X is a vertex cover of G, and an independent set of H ?

We call the set X as a *conflict-free vertex cover*. Next, we show that CF-VC can be solved as fast as the classical VERTEX COVER problem. Towards this, we present a polynomial time reduction from CF-VC to MIN ONES 2-SAT which preserves both the parameter k and the number of variables n. For a 2-CNF formula Φ and a satisfying assignment τ of Φ, if τ sets ℓ variables in Φ to 1, then we say that weight of τ is ℓ. The MIN ONES 2-SAT problem is defined formally as follows.

MIN ONES 2-SAT

Input: A 2-CNF formula Φ and an integer k.

Question: Does there exist a satisfying assignment τ of Φ where at most k variables are set to 1?

Construction 2. *Given a 2-CNF formula Φ, let $V(\Phi)$ and $C(\Phi)$ denote the set of variables and clauses of Φ, respectively. Given an instance (G,H,k) of* CF-VC*, we construct an instance (Φ,k) of* MIN ONES 2-SAT *as follows. For every edge $uv \in E(G)$, introduce a clause $(u \vee v)$ and for every edge $uv \in E(H)$, introduce a clause $(\bar{u} \vee \bar{v})$ in Φ. More precisely, given the graphs G and H, the* CF-VC *is formulated as the following instance of* MIN ONES 2-SAT.

$$\Phi = (\bigwedge_{uv \in E(G)} (u \vee v)) \bigwedge (\bigwedge_{uv \in E(H)} (\bar{u} \vee \bar{v})).$$

Lemma 10. *(G,H,k) is a* yes*-instance of* CF-VC *if and only if (Φ,k) is a* yes*-instance of* MIN ONES 2-SAT. *Furthermore, $|V(\Phi)| = |V(G)| = |V(H)|$.*

Proof. For the forward direction, let (G,H,k) be a yes-instance of CF-VC and let X be its solution of size at most k. We construct a truth assignment τ of Φ as follows. If $x \in X$ then $\tau(x) = 1$, otherwise it is 0. Clearly, this satisfies the formula Φ and it is of weight at most k. Conversely, let τ be a satisfying assignment of Φ of weight at most k. We construct a set X as follows. If $\tau(u) = 1$, add the vertex u to X. For the clause $(u \vee v)$, at least one of $\tau(u)$ or $\tau(v)$ is 1. This ensures that every edge of G is incident to some vertex $u \in X$. For the clause $(\bar{u} \vee \bar{v})$, at least one of $\tau(u)$ or $\tau(v)$ is 0. This ensures that $H[X]$ is edgeless. Clearly, the size of X is at most k. This completes the proof. □

Using Lemma 10, we obtain the following lemma.

Lemma 11. *Let G be a graph and H be a conflict graph of G. Then, in polynomial time we can test whether there exists a conflict-free vertex cover of the pair (G,H).*

Proof. We construct a 2-SAT formula, Φ using Construction 2. Similar, to the arguments of the proof of Lemma 10, we can show that (G,H) has a conflict-free vertex cover (of any size) if and only if Φ has a satisfying assignment. We know that 2-SAT is polynomial time solvable [73]. This implies that, testing whether the pair (G,H) has a conflict-free vertex cover can be done in polynomial time. □

Misra et al. [89] have shown that MIN ONES 2-SAT is equivalent to VERTEX COVER in both the parameterized and optimization settings. In Lemma 10, we have shown that CF-VC is equivalent to MIN ONES 2-SAT. In our reduction from CF-VC to MIN ONES 2-SAT we preserve the parameter and the number of vertices in the graphs in CF-VC is equal to the number of variables in the formula in MIN ONES 2-SAT. This implies that the best known algorithms for VERTEX COVER can be employed to CF-VC incurring only an additional polynomial cost. Using results from [27, 60, 110], we obtain the following.

Theorem 3.3.1. CF-VC *admits a* $2k$*-vertex kernel, a factor* 2*-approximation algorithm, an* $\mathcal{O}^\star(1.2738^k)$ FPT *algorithm, and a* $\mathcal{O}^\star(1.1996^n)$ *exact algorithm.*

It is well known that VERTEX COVER is NP-complete in general and polynomial time solvable for graphs with maximum degree at most two. In the following, we prove that, $(\mathscr{P}_3,\star)$-CF-VC is NP-complete, where $\mathscr{P}_3$ denotes the class of disjoint union of P_3s (P_ℓ denotes path on ℓ vertices). Towards this, we present a polynomial time reduction from the MIN ONES 2-SAT problem to $(\mathscr{P}_3,\star)$-CF-VC. Let (Φ,k) be an instance of MIN ONES 2-SAT and let $V(\Phi)$ and $C(\Phi)$ denote the set of variables and clauses of Φ, respectively. Given an instance (Φ,k) of MIN ONES 2-SAT, we construct an instance (G,H,k') of $(\mathscr{P}_3,\star)$-CF-VC as follows. For each variable $x \in V(\Phi)$, introduce three vertices x_1, x_2 and x_3 in G and H such that x_1 and x_2 corresponds to the positive literal and x_3 corresponds to the negative literal of x. Add edges x_1x_3 and x_2x_3 in both G and H. Note that G is collection of P_3's and there is a P_3, (x_1,x_3,x_2) in G corresponding to each variable $x \in V(\Phi)$. For each clause $(l_1 \vee l_2) \in C(\Phi)$, where l_1,l_2 are literals of the variable x_1,x_2, respectively, add edges between the vertices corresponding to $\bar{l}_1$ literal of x_1 and the

vertices corresponding to $\bar{l}_2$ literal of x_2 in H. Let $|V(\Phi)| = n$. Set $k' = n+k$. Clearly, this construction can be carried out in polynomial time.

Lemma 12. (Φ, k) *is a* yes*-instance of* MIN ONES 2-SAT *if and only if* $(G, H, k' = n+k)$ *is a* yes*-instance of* $(\mathscr{P}_3, \star)$-CF-VC.

Proof. In the forward direction, let (Φ, k) be a yes-instance of MIN ONES 2-SAT and let τ be a satisfying assignment of Φ of weight at most k. We construct a set X corresponding to the truth assignment τ as follows. If $\tau(x) = 1$, add vertices x_1, x_2 to X otherwise add x_3 to X. Clearly X is a vertex cover of G. We will now prove that $H[X]$ is edgeless. Suppose for a contradiction that $H[X]$ is not edgeless and suppose there exists an edge uv in $H[X]$. By the construction of H, either u and v corresponds to the positive and the negative literal of a variable x in $V(\Phi)$, respectively, or there exists a clause $\bar{u} \vee \bar{v}$ in $C(\Phi)$, which contradicts the fact that τ is a truth assignment of Φ. Since weight of τ is at most k, at least $n-k$ variables are set to 0. This implies that $|X| \leq 2k+n-k = n+k$.

Conversely, let $(G, H, k' = n+k)$ be a yes-instance of $(\mathscr{P}_3, \star)$-CF-VC and let X be its minimal solution of size at most $n+k$. We assign a truth assignment τ to Φ corresponding to the conflict-free vertex cover X as follows. If for a P_3 corresponding to a variable x both the endpoints are in X, set $\tau(x) = 1$, otherwise 0. We now show that τ is a satisfying assignment of Φ. Suppose for a contradiction there exists a clause $(x \vee y) \in C(\Phi)$ that is not satisfied by τ. This implies that $\tau(\bar{x}) = \tau(\bar{y}) = 1$. By the construction of H, there exists an edge uv in $H[X]$ where u and v corresponds to $\bar{x}$ and $\bar{y}$, respectively, which is a contradiction to the assumption that X is conflict-free vertex cover. Since X is minimal, for each P_3 either both the endpoints or the central vertex is in X. Therefore, from at most k number of P_3's both the endpoints are selected in X. This implies that the weight of τ is at most k. □

Using construction and Lemma 12 we obtain the following result.

Theorem 3.3.2. $(\mathscr{P}_3, \star)$-CF-VC *is NP-complete.*

However, certain special cases of CF-VC are polynomial time solvable.

Theorem 3.3.3. *($\mathscr{G}_{\leq 1}, \star$)-CF-VC is polynomial time solvable, where $\mathscr{G}_{\leq 1}$ is class of graphs with degree at most one.*

Proof. Let G be a graph in the graph class $\mathscr{G}_{\leq 1}$ and H be a conflict graph. We construct a 2-SAT formula, Φ as described in Construction 2. We will prove that using a satisfying assignment of Φ, it is possible to obtain an optimal conflict-free vertex cover, X, of (G,H). We know that 2-SAT is polynomial time solvable [73]. If Φ is not satisfiable then we return that the pair (G,H) has no conflict-free vertex cover. So, now we assume that Φ is satisfiable. Let τ be a truth assignment of Φ. We construct a set X as follows. If $\tau(x) = 1$, add x to X. Clearly, in the satisfying assignment of Φ, for each edge $uv \in E(G)$ at least one of $\tau(u)$ or $\tau(v)$ is 1 and for each edge $uv \in H$ at least one of $\tau(u)$ or $\tau(v)$ is 0. In this manner, X is a conflict-free vertex cover. Now the aim is to minimize the size of resulting X. Trivially, if for any edge $uv \in E(G)$ both the endpoints of it belong to X, then we can safely delete one of them. Since, we need to pick at least one vertex from each edge, we have that the resulting X' is of the minimum size. Since it is a subset of a conflict-free vertex cover, we have that X' is also a conflict-free vertex cover. This concludes the proof. □

Theorem 3.3.4. *($\mathscr{M}, \mathscr{M}$)-CF-VC is polynomial time solvable, where $\mathscr{M}$ is class of graphs with perfect matching.*

Proof. Let G, H (conflict graph) be the graphs in the graph class $\mathscr{M}$. We construct a 2-sat formula, Φ as described in Construction 2. If Φ is not satisfiable, then we return that the pair (G,H) has no conflict vertex cover. So, now we assume that Φ is satisfiable. Let τ be a truth assignment of Φ. We construct a set X as follows. If $\tau(x) = 1$, add x to X. Clearly, in the satisfying assignment of Φ, for each edge $uv \in E(G)$ at least one of $\tau(u)$ or $\tau(v)$ is 1 and for each edge $uv \in H$ at least one of $\tau(u)$ or $\tau(v)$ is 0. In this manner, X is a conflict-free vertex cover. Since G has a perfect matching, at least $n/2$ variables are set to

true and since H also has a perfect matching at least $n/2$ variables are set to false. This implies that, exactly $n/2$ variables are set to true. Hence, the corresponding conflict-free vertex cover is of size $n/2$. This shows that, if there exists a satisfying assignment τ of Φ, then the resulting X is optimal. □

3.3.2 CONFLICT FREE d-HITTING SET

The HITTING SET (HS) problem is a generalization of VERTEX COVER. In HS, given a family of sets, $\mathscr{S}$, over a ground set U of n elements and an integer k, the objective is to check whether there exists a subset of U of size at most k which intersects (or hits) every set of the family. In d-HITTING SET (d-HS), every set in the family has at most d elements. By $\binom{U}{\leq d}$, we denote the set of all the subsets of U of size at most d. The conflict-free version of d-HS, namely CONFLICT FREE d-HITTING SET (d-CF-HS), is defined as follows.

CONFLICT FREE d-HITTING SET (d-CF-HS) **Parameter:** k
Input: A family $(U, \mathscr{S})$, $\mathscr{S} \subseteq \binom{U}{\leq d}$, a conflict graph H with $V(H) = U$, and an integer k.
Question: Does there exist a subset $X \subseteq U$ of size at most k, such that $S_i \cap X \neq \phi$ for all $S_i \in \mathscr{S}$ and $H[X]$ is edgeless?

We call the set X as a *conflict-free hitting set*. Observe that, CF-FINITE Π-VD is a special case of d-CF-HS and thus, a faster algorithm for d-CF-HS implies a faster algorithm for CF-FINITE Π-VD. In the following, we give an $\mathscr{O}^\star((d-0.7262)^k)$ FPT algorithm for d-CF-HS using iterative compression and the algorithm for CF-VC given in Theorem 3.3.1.

Firstly, we propose an algorithm for 3-CF-HS, later we generalize it for d-CF-HS. Given an instance $(\mathscr{S}, U, k)$ of 3-CF-HS, the algorithm first checks whether there exists a hitting set of $(\mathscr{S}, U, k)$ of size at most k using FPT algorithm for 3-HS in $\mathscr{O}^\star(2.0755^k)$

time [107]. If it returns *n*o, then $(\mathscr{S}, U, H, k)$ does not have a conflict-free hitting set of size at most k and the algorithm returns *n*o. Otherwise, it returns a hitting set, Y, of size at most k. The algorithm now passes $(\mathscr{S}, U, H, Y, k)$ to a search routine, described below, which either finds a conflict-free hitting set, X, of size at most k or returns *n*o.

Next, we describe the search routine. The search routine iterates over all the choices of the set $W = Y \cap X$, where $0 \leq |W| \leq k$. In an iteration, if $H[W]$ is not edgeless then the routine proceeds to the next choice of W. Otherwise, let $N = Y \setminus W$ and $\mathscr{C} \subseteq \mathscr{S}$ be the set of all the sets which are hit by W. Let $\mathscr{S}' = \mathscr{S} \setminus \mathscr{C}$ and $U' = \cup_{S \in \mathscr{S}'} S$. The aim is to find a conflict-free hitting set, $Z \subseteq (U' \setminus N)$ of $(\mathscr{S}', U', H, k')$ of size at most $k' = k - |W|$ such that $H[Z \cup W]$ is edgeless. Towards this, the algorithm first marks all the vertices in $S \in \mathscr{S}'$ which are either in N or neighbors of W in H, to signify that these vertices cannot be picked in the solution. Since, the marked vertices cannot be part of any solution, if a set in $\mathscr{S}'$ contains all marked vertices, then the algorithm returns *n*o and stop. Otherwise, deletes all the marked vertices from each $S \in \mathscr{S}'$. It is to be noted here that the problem is reduced to 2-CF-HS and thus to CF-VC which can be solved in $\mathscr{O}^\star(1.2738^k)$ time. Let us analyze the running time of this algorithm. For a given choice of the set W; $|W| = i$, $0 \leq i \leq k$, the search routine takes $\mathscr{O}^\star(1.2738^{k-i})$ time to find the solution. The running time of search routine taken over all choices of the set W is thus $\sum_{i=0}^{k} \binom{k}{i} (1.2738)^{k-i} = \mathscr{O}^\star(2.2738^k)$. One can easily extend this algorithm to an algorithm for d-CF-HS, by using an algorithm for d-HS, which runs in time $\mathscr{O}^\star((d-0.7262)^k)$ [107] and by using $(d-1)$-CF-HS, as a subroutine. Hence, we obtain the following result.

Theorem 3.3.5. *The* d-CF-HS *problem can be solved in* $\mathscr{O}^\star(((d-1)+0.2738)^k) = \mathscr{O}^\star((d-0.7262)^k)$ *time.*

3.3.3 Conflict Free Split Vertex Deletion

In this section, we give a faster FPT algorithm for conflict-free version of Split Vertex Deletion(SVD) problem, namely Conflict Free Split Vertex Deletion (CF-

SVD). A *split graph* is a graph whose vertices can be partitioned into a clique and an independent set. The problem CF-SVD is defined formally as follows.

CONFLICT FREE SPLIT VERTEX DELETION (CF-SVD) **Parameter:** k
Input: A graph G, a conflict graph H, and an integer k.
Question: Does there exist a subset $S \subseteq V(G)$ of size at most k, such that $G-S$ is a split graph and $H[S]$ is an independent set?

A family of split graphs has $2K_2, C_4, C_5$ as finite forbidden induced subgraph [50] and thus by Theorem 3.2.1, CF-SVD admits $\mathcal{O}^\star(5^k)$ algorithm. In the following, we describe a faster algorithm for CF-SVD. We first state the following result, that will be used later.

Proposition 10. [Theorem 1.3,[32]] *For any graph G on n-vertices, there exists a family $\mathscr{P}$ of partitions (V_C, V_I) of the vertex set $V(G)$, of size at most $4(2n)^{2\lfloor \log n \rfloor}$, such that for any set $X \subseteq V(G)$, $G[X]$ is a split graph, and any partition (X_C, X_I) of X such that $G[X_C]$ is a clique and $G[X_I]$ is an independent set, there exists a partition $(V_C, V_I) \in \mathscr{P}$ such that $X_C \subseteq V_C$ and $X_I \subseteq V_I$. Such a family can be generated in time $\mathcal{O}^*(n^{\mathcal{O}(\log n)})$ time and polynomial space.*

Next, we prove the following result using Proposition 10.

Theorem 3.3.6. *If* CF-VC *on n vertex graph, parameterized by the solution size k can be solved in time $T(k,n)$, then the* CF-SVD *problem can be solved in time $\mathcal{O}^\star(T(k,n)k^{\mathcal{O}(\log k)})$.*

Proof. Let (G, H, k) be an instance of CF-SVD. We first apply the polynomial time kernelization algorithm given in Theorem 3.2.4 to find a $\mathcal{O}(k^5)$ vertex kernel. Let (G', H', k') be the reduced instance. Next, we use the partitioning algorithm given by Proposition 10 to find a family of partitions (V_C, V_I) for the reduced graph G'. Since, number of vertices are bounded by $\mathcal{O}(k^5)$, by Proposition 10, enumerating the family of such partitions will take $\mathcal{O}^*(k^{\mathcal{O}(\log k)})$ time and polynomial space. For a partition (V_C, V_I) of $V(G')$, the goal is to find a set $X \subseteq V(G')$ such that $G'[V_C \cap X]$ is a clique and $G'[V_I \cap X]$ is an independent set, and if $S = V(G') \setminus X$, then $|S| \leq k$ and $H'[S]$ is edgeless. Observe that, if there are vertices

$u, v \in V_C$ which are not adjacent, then we have to either delete u or v to make $G'[V_C]$ a clique. Similarly, if there are vertices $u, v \in V_I$ which are adjacent, then we have to either delete u or v to make $G'[V_I]$ an independent set. Also, the deleted vertices should form an independent set in the graph H'. Let E_C be the edges in complement graph of $G'[V_C]$ and E_I be the edges in graph $G'[V_I]$. Our problem now reduces to an instance of CF-VC in the graph with vertex set $V(G')$ and the edge set $E_C \cup E_I$, with the conflict graph H' and parameter k. Therefore, for a fixed partition the problem can be solved in the same time as the CF-VC problem. This concludes the proof. □

We obtain the following corollary using Theorems 3.3.1 and 3.3.6.

Corollary 6. CF-SVD *can be solved in* $\mathcal{O}^\star(1.2738^k k^{\mathcal{O}(\log k)})$ *time and polynomial space.*

3.3.4 Conflict Free Feedback Vertex Set in Tournaments

In this section, we give a faster FPT algorithm for conflict-free version of Feedback Vertex Set in Tournaments (FVST), namely Conflict Free Feedback Vertex Set in Tournaments (CF-FVST). Our FPT algorithm uses iterative compression technique. A simple algorithm for FVST is based on the following well known result.

Lemma 13. [29] *Let G be a tournament. G has a directed cycle if and only if G has a directed triangle. Otherwise, G is acyclic and it has a unique topological ordering.*

This lemma will be useful to prove further results in this section. CF-FVST is defined formally as follows.

Conflict Free Feedback Vertex Set in Tournaments (CF-FVST)
Input: A tournament G, a conflict graph H, and an integer k.
Parameter: k
Question: Does there exist a subset $X \subseteq V(G)$ such that $\|X\| \leq k$, $G - X$ is directed acyclic graph (dag) and X is an independent set in H?

A set X is called as a *conflict-free fvst*. Next, we describe our algorithm. The algorithm first checks whether there exists a feedback vertex set of G of size at most k using FPT algorithm for FVST in $\mathcal{O}^\star(1.618^k)$ time [74]. If it returns *n*o, then, (G,H,k) cannot have conflict-free fvst of size at most k and the algorithm returns *n*o. Otherwise, it returns a feedback vertex set, F of size at most k. The algorithm now passes (G,H,F,k) to a search routine, described below, which either finds conflict-free fvst, X of (G,H,k) of size at most k or returns *n*o.

Next, we describe the search routine. It iterates over all choices of the set $Y \subseteq F$ where $0 \leq |F| \leq k$. Let $N = F \setminus Y$ and $W = V(G) \setminus F$. The routine rejects Y if either $H[Y]$ is not edgeless or $G[N]$ is not acyclic. Now, the objective is to find a subset $Z \subseteq W$ such that $G[(W \cup N) \setminus Z]$ is acyclic and $H[Y \cup Z]$ is edgeless. The algorithm marks all the neighbors of Y in W in the graph H to signify that they cannot be picked into a solution. Let R be the set of marked vertices. In the following, we state some reduction rules.

Reduction Rule 3. *If $G[W \cup N]$ has a directed triangle (x,y,z) such that $x,y,z \in (N \cup R)$, return* n*o.*

Reduction Rule 4. *If $G[W \cup N]$ has a directed triangle (x,y,z) with exactly one vertex from $W \setminus R$, say x, delete x and reduce parameter k by* 1.

It is easy to see that Reduction rules 3 and 4 are safe. The algorithm first applies Reduction Rules 3 and 4 exhaustively. Since $G[N]$ and $G[W]$ are acyclic, by Lemma 13, both have unique topological ordering. Nect, we define an ordering π of vertices of $G[W]$. Let σ and ρ be the topological orderings of $G[N]$ and $G[W]$, respectively. Since Reduction Rule 4 is no longer applicable, $G[N \cup \{v\}]$ is acyclic for all $v \in W$. This implies that, $G[N \cup \{v\}]$ has a unique topological ordering σ', by Lemma 13. Observe that, σ' preserves σ, that is, σ' restricted to N is same as the ordering σ. Let $\sigma = \{u_1, \ldots, u_q\}$. Then, there exists a unique integer $p(v)$ such that for all $i < p(v)$, there is an arc (u_i, v) in G and for all $i \geq p(v)$, there is an arc (v, u_i) in G. Observe that, $p(v)$ is uniquely defined for all $v \in W$ and $p(v) \in [q+1]$. Next, we construct an ordering π of W as follows: for all $u, v \in W$, the

position of u is before v in π iff $p(u) < p(v)$, or $p(u) = p(v)$ and the position of u is before v in ρ. Clearly, there is a cycle in $G[N \cup W]$ iff there exists an arc uv in $G[W]$ such that $\rho(u) \prec \rho(v)$ and $\pi(u) \succ \pi(v)$. This observation leads to following lemma.

Lemma 14 ([42], Lemma 4.1)**.** *Consider that, $B \subseteq W$. $G[B \cup N]$ is acyclic iff the vertices of B form a common subsequence of ρ and π.*

Next, given a set U and two sequences S_1, S_2 of U, where every element of U occurs exactly once in S_1 and S_2, we present an algorithm to find all maximal subsequences in S_1, S_2. We assign a pair (τ_u, γ_u) to each element $u \in U$, where τ_u and γ_u denote position of u in S_1 and S_2, respectively. Next, we construct a digraph $D = (V, A)$, where $V(D) = \{(\tau_u, \gamma_u), \forall u \in U\}$. We add an arc $((\tau_u, \gamma_u), (\tau_v, \gamma_v))$ if $\tau_u < \tau_v$ and $\gamma_u < \gamma_v$. Clearly, D is directed acyclic graph. Next, we run the well known depth-first search (DFS) algorithm on the digraph D starting from the smallest vertex in lexicographic ordering of vertices. In DFS, we will always visit the smallest vertex first according to lexicographic ordering. DFS returns all maximal paths in D in polynomial time. Suppose that, $((\tau_u, \gamma_u), (\tau_v, \gamma_v), \cdots, (\tau_w, \gamma_w))$ is a maximal path in D then $u, v, \cdots, w$ is a maximal subsequence in S_1 and S_2. Next, we prove a lemma which formalizes this statement.

Lemma 15. *There exists a maximal common subsequence $u, v, \cdots, w$ of S_1 and S_2 if and only if $((\tau_u, \gamma_u), (\tau_v, \gamma_v), \cdots, (\tau_w, \gamma_w))$ is a maximal path in D.*

Proof. Suppose that, there exists a common subsequence $u, v, \cdots, w$ of S_1 and S_2. This implies that, $\tau_u \prec \tau_v \prec \cdots \prec \tau_w$ and $\gamma_u \prec \gamma_v \prec \cdots \prec \gamma_w$. By the construction of D, there is a path between $((\tau_u, \gamma_u), (\tau_v, \gamma_v), \cdots, (\tau_w, \gamma_w))$ in D. Conversely, let there exists a maximal path between $((\tau_u, \gamma_u), (\tau_v, \gamma_v), \cdots, (\tau_w, \gamma_w))$ in D then similar to argument as above $u, v, \cdots, w$ is a common subsequence of S_1 and S_2. □

For sequences of W with respect to the orderings ρ and π, we find a longest subsequence Z among all maximal subsequences such that $G[(N \cup Z)]$ is directed acyclic graph and $H[Y \cup (W \setminus Z)]$ is edgeless. Using Lemmas 14 and 15, we obtain the following result.

Theorem 3.3.7. CF-FVST *can be solved in* $\mathcal{O}^\star(2^k)$ *time.*

3.4 Conclusion

In this chapter, we introduced a new variant, called the conflict-free version, of classical vertex deletion problems that are studied in graph algorithms. We studied these problems in the realm of Parameterized Complexity and obtained several results that classify the complexity of these problems in various graph classes. Our work opens up a whole new area of research in obtaining dichotomy results. For every property Π, where CONFLICT FREE Π-VERTEX DELETION is $\mathsf{W}[1]$-hard, it is a natural question to ask for which family of graphs H does the problem becomes FPT. As a concrete question in this direction, for which graph classes $\mathscr{G}, \mathscr{H}$, the problems $(\mathscr{G}, \mathscr{H})$-CF-FVS and $(\mathscr{G}, \mathscr{H})$-CF-OCT admit FPT algorithms and polynomial kernels.

Chapter 4

Conflict-Free Feedback Vertex Set: A Parameterized Dichotomy

4.1 Introduction

FEEDBACK VERTEX SET (FVS) is one of the classical NP-hard problems that has been subjected to intensive study in algorithmic paradigms that are meant for coping with NP-hard problems, and particularly in the realm of Parameterized Complexity. Recall that, in this problem, given a graph G and an integer k, the objective is to decide if there is $S \subseteq V(G)$ of size at most k, such that $G - S$ is a forest. FVS has received a lot of attention in the realm of Parameterized Complexity. This problem is known to be in FPT, and the best known algorithm for it runs in time $\mathcal{O}^\star(2.7^k)$ [77]. Several variant and generalizations of FEEDBACK VERTEX SET such as WEIGHTED FEEDBACK VERTEX SET [3, 26], INDEPENDENT FEEDBACK VERTEX SET [2, 90], CONNECTED FEEDBACK VERTEX SET [91], and SIMULTANEOUS FEEDBACK VERTEX SET [4, 20] have been studied from the viewpoint of Parameterized Complexity.

In Chapter 3 we defined a generalization of well-studied vertex deletion problems

– in particular for FVS. Recall that, the CF-FEEDBACK VERTEX SET (CF-FVS, for short) problem takes as input graphs G and H, and an integer k, and the objective is to decide if there is a set $S \subseteq V(G)$ of size at most k, such that $G - S$ is a forest and S is an independent set in H. The graph H is also called a *conflict graph*. Observe that the CF-FVS problem generalizes classical graph problems, FEEDBACK VERTEX SET and INDEPENDENT FEEDBACK VERTEX SET. In Chapter 3 we defined CF-FVS by fixing a family $\mathscr{F}$ from which the conflict graph H is allowed to belong. Thus, for every fixed $\mathscr{F}$ we get a new CF-FVS problem. We recall the definition here.

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

In Chapter 3, we showed that $\mathscr{F}$-CF-FVS is W[1]-hard when $\mathscr{F}$ is a family of all graphs and admits FPT algorithm when the input graph H is from the family of d-degenerate graphs and the family of nowhere dense graphs. The most natural question that arises here is the following.

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

In this chapter we start by exploring Question 1. We obtain a complete dichotomy result on the Parameterized Complexity of the problem $\mathscr{F}$-CF-FVS (for hereditary $\mathscr{F}$) in terms of another well-studied problem, namely, the INDEPENDENT SET problem – the wall of intractability. Towards stating our results in the chapter, we start by defining the problem $\mathscr{F}$+CLUSTER IS, which is of independent interest. A *cluster graph* is a graph formed from the disjoint union of complete graphs (or cliques).

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

We note that $\mathscr{F}$+CLUSTER IS is the INDEPENDENT SET problem on the edge union of two graphs, where one of the graphs is from the family of graphs $\mathscr{F}$ and the other one is a cluster graph. Here, additionally we know the partition of edges into two sets, E_1 and E_2 such that the graph induced on E_1 is in $\mathscr{F}$ and the graph induced on E_2 is a cluster graph. We note that $\mathscr{F}$+CLUSTER IS has been studied in the literature for $\mathscr{F}$ being the family of interval graphs (with no restriction on the number of clusters) [106]. They showed the problem to be FPT. Recently, Bentert et al. [10] generalized the result from interval graphs to chordal graphs. This problem arises naturally in the study of scheduling problems. We refer the readers to [106, 10] for more details on the application of $\mathscr{F}$+CLUSTER IS.

In this chapter, we show that $\mathscr{F}$-CF-FVS is in FPT if and only if $\mathscr{F}$+CLUSTER IS is in FPT, where $\mathscr{F}$ is a family of hereditary graphs. We obtain a complete characterization of when the $\mathscr{F}$-CF-FVS problem is in FPT, for hereditary graph families. To prove the forward direction, that is, showing that $\mathscr{F}$+CLUSTER IS is in FPT implies $\mathscr{F}$-CF-FVS is in FPT, we design a branching based algorithm, which at the base case generates instances of $\mathscr{F}$+CLUSTER IS, which is solved using the assumed FPT algorithm for $\mathscr{F}$+CLUSTER IS. Thus, we give "fpt-turing-reduction" from $\mathscr{F}$-CF-FVS to $\mathscr{F}$+CLUSTER IS. It is worth to note that there are very few known reductions of this nature. To show that $\mathscr{F}$-CF-FVS is in FPT implies that $\mathscr{F}$+CLUSTER IS is in FPT, we give an appropriate reduction from $\mathscr{F}$+CLUSTER IS to $\mathscr{F}$-CF-FVS, which proves the statement. We note that our result that $\mathscr{F}$-CF-FVS is in FPT implies $\mathscr{F}$+CLUSTER IS is in FPT, holds for all families of graphs.

Next, we consider two families of graphs. We first design FPT algorithm for the

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

We note that $K_{i,j}$-free graphs have at most $n^{2-\varepsilon}$ edges, where n is the number of vertices in the input graph and $\varepsilon = \varepsilon(i,j) > 0$ [103, 56]. We complement our FPT result on $K_{i,j}$-free+CLUSTER IS with the W[1]-hardness result of the $\mathscr{F}$+CLUSTER IS problem when $\mathscr{F}$ is the family of graphs with at most $n^{2-\varepsilon}$ edges. This result is obtained by giving an appropriate reduction from the problem MULTICOLORED BICLIQUE, which is known to be W[1]-hard [29, 48]. We also show that the $\mathscr{F}$+CLUSTER IS problem is W[1]-hard when $\mathscr{F}$ is the family of bipartite graphs. Again, this result is obtained via a reduction from MULTICOLORED BICLIQUE.

4.2 Preliminaries

In this section, we state some basic definitions and terminologies from Graph Theory that are used in this chapter. For the graph related terminologies which are not explicitly defined here, we refer the reader to the book of Diestel [40].

Graphs Consider a graph G. By $V(G)$ and $E(G)$ we denote the set of vertices and edges in G, respectively. When the graph is clear from the context, we use n and m to denote the number of vertices and edges in the graph, respectively. For $X \subseteq V(G)$, by $G[X]$ we denote the subgraph of G with vertex set X and edge set $\{uv \in E(G) \mid u,v \in X\}$. Moreover, by $G-X$ we denote graph $G[V(G) \setminus X]$. For $v \in V(G)$, $N_G(v)$ denotes the set $\{u \mid uv \in E(G)\}$, and $N_G[v]$ denotes the set $N_G(v) \cup \{v\}$. By $\deg_G(v)$ we denote the size of $N_G(v)$. A *path*

$P = (v_1, \ldots, v_n)$ is an ordered collection of vertices, with endpoints v_1 and v_n, such that there is an edge between every pair of consecutive vertices in P. A *cycle* $C = (v_1, \ldots, v_n)$ is a path with the edge v_1v_n. Consider graphs G and H. We say that G is an *H-free* graph if no subgraph of G is isomorphic to H. For $u, v \in V(G) \cap V(H)$, we say that u and v are in *conflict* in G with respect to H if $uv \in E(H)$.

A *clique* is a subgraph of an undirected graph such that every two distinct vertices in it are adjacent. A *connected component* of an undirected graph is a (vertex) maximal induced subgraph in which every two vertices are connected by a path. If a graph has only one connected component then it is called a *connected graph*. A graph is a *cluster* graph if each of its connected components are cliques. For $k \in \mathbb{N}$, a *k-cluster* graph is cluster graph with exactly k connected components. Let $\mathscr{C}$ be the set of connected components in cluster graph. We define vertex set of $\mathscr{C}$ as follows: $V(\mathscr{C}) = \cup_{C \in \mathscr{C}'} V(C)$. A graph G is a *complete bipartite* graph if its vertex set can be partitioned into two disjoint (independent) sets X and Y, such that $E(G) = \{xy \mid x \in X, y \in Y\}$. For $x, y \in \mathbb{N}$, by K_{xy} we denote the complete bipartite graph on $x+y$ vertices which admits a vertex bipartition into sets X and Y of sizes x and y, respectively, such that $E(K_{xy}) = \{xy \mid x \in X, y \in Y\}$.

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

4.3 W-hardness of $\mathscr{F}$-CF-FVS Problems

This section is devoted to showing W-hardness results for $\mathscr{F}$-CF-FVS problems for certain graph classes, $\mathscr{F}$. In Section 4.3.1, we show one direction of our dichotomy result. That is, if for a family of graphs $\mathscr{F}$, $\mathscr{F}$+CLUSTER IS is not in FPT when parameterized by the size

of solution then $\mathscr{F}$-CF-FVS is also not in FPT when parameterized by the size of solution. This result is obtained by giving a parameterized reduction from $\mathscr{F}$+CLUSTER IS to $\mathscr{F}$-CF-FVS. Next, we show that the problem $\mathscr{F}$-CF-FVS is W[1]-hard, when parameterized by the size of solution, where $\mathscr{F}$ is the family of bipartite graphs (Section 4.3.2) or the family of graphs with sub-quadratic number of edges (Section 4.3.3). These results are obtained by giving an appropriate reduction from the problem MULTICOLORED BICLIQUE, which is known to be W[1]-hard [29, 48].

4.3.1 $\mathscr{F}$+CLUSTER IS to $\mathscr{F}$-CF-FVS

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

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

Lemma 16. *(G,H,k) is a* yes*-instance of $\mathscr{F}$+CLUSTER IS if and only if (G',H',k') is a* yes*-instance of $\mathscr{F}$-CF-FVS.*

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

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

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

Theorem 4.3.1. *For a family of graphs $\mathscr{F}$, if $\mathscr{F}+$CLUSTER IS is not in* FPT *when parameterized by the solution size, then $\mathscr{F}$-*CF-FVS *is also not in* FPT *when parameterized by the solution size.*

Proof. Follows from the construction of instance (G', H', k') of $\mathscr{F}$-CF-FVS from the given instance (G, H, k) of $\mathscr{F}+$CLUSTER IS with $H' = G$ and Lemma 16. □

4.3.2 W[1]-hardness on Bipartite Graphs

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

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

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

Lemma 17. *$(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ is a* yes*-instance of* MULTICOLORED BICLIQUE *if and only if (G', H', k') is a* yes*-instance of $\mathscr{B}$-*CF-FVS.

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

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

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

Theorem 4.3.2. *$\mathscr{B}$-*CF-FVS *parameterized by the solution size is* W[1]*-hard, where $\mathscr{B}$ is the family of bipartite graphs.*

Proof. Follows from the construction of instance (G', H', k') of $\mathscr{B}$-CF-FVS from the given instance $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ of MULTICOLORED BICLIQUE, Lemma 17, and

W[1]-hardness of MULTICOLORED BICLIQUE [29, 48]. □

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

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

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

Lemma 18. *$(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ is a* yes*-instance of* MULTICOLORED BICLIQUE *if and only if (G', H', k') is a* yes*-instance of $\mathscr{F}_\varepsilon$-*CF-FVS.

Proof. In the forward direction, let $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ be a yes-instance of MULTICOLORED BICLIQUE, and S be one of its solution. Let $x \in X$ be an arbitrarily chosen vertex from X. We will show that $S \cup \{x\}$ is a solution to $\mathscr{F}_\varepsilon$-CF-FVS in (G', H', k'). Since S is a solution to MULTICOLORED BICLIQUE in $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$, for each $i \in [k]$, $|S \cap A_i| = 1$ and $|S \cap B_i| = 1$. Since G' comprises of vertex disjoint cycles corresponding to sets in A_i and B_i, and a cycle induced on vertices in X, we have that $S \cup \{x\}$ intersects

every cycle in G'. By the construction of H' it follows that $S \cup \{x\}$ is an independent set in H'. This concludes the proof of forward direction.

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

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

Theorem 4.3.3. *For* $0 < \varepsilon < 1$, $\mathscr{F}_\varepsilon$-CF-FVS *parameterized by the solution size is* W[1]-hard.

Proof. Follows from the construction of instance (G', H', k') of $\mathscr{F}_\varepsilon$-CF-FVS from the given instance $(G, A_1, \cdots, A_k, B_1, \cdots, B_k)$ of MULTICOLORED BICLIQUE, Lemma 18, and W[1]-hardness of MULTICOLORED BICLIQUE [29, 48]. □

4.4 FPT algorithms for $\mathscr{F}$-CF-FVS for Restricted Conflict Graphs

For a hereditary (closed under taking induced subgraphs) family of graphs $\mathscr{F}$, we show that if $\mathscr{F}$+CLUSTER IS is FPT, then $\mathscr{F}$-CF-FVS is FPT. Throughout this section, whenever we refer to a family of graphs, it will refer to a hereditary family of graphs. To prove our result, for a family of graphs $\mathscr{F}$, for which $\mathscr{F}$+CLUSTER IS is FPT, we will design an FPT algorithm for $\mathscr{F}$-CF-FVS, using the (assumed) FPT algorithm for $\mathscr{F}$+CLUSTER IS. We note that this gives us a Turing parameterized reduction from $\mathscr{F}$-CF-FVS to $\mathscr{F}$+CLUSTER IS. Our algorithm will use the technique of compression together with branching. We note that the method of iterative compression was first introduced by Reed,

Smith, and Vetta [102], and in our algorithm, we (roughly) use only the compression procedure from it.

In the following, we let $\mathscr{F}$ to be a (fixed hereditary) family of graphs, for which $\mathscr{F}$+CLUSTER IS is in FPT. Towards designing an algorithm for $\mathscr{F}$-CF-FVS, we define another problem, which we call $\mathscr{F}$-DISJOINT CONFLICT FREE FEEDBACK VERTEX SET (to be defined shortly). Firstly, we design an FPT algorithm for $\mathscr{F}$-CF-FVS using an assumed FPT algorithm for $\mathscr{F}$-DISJOINT CONFLICT FREE FEEDBACK VERTEX SET. Secondly, we give an FPT algorithm for $\mathscr{F}$-DISJOINT CONFLICT FREE FEEDBACK VERTEX SET using the assumed algorithm for $\mathscr{F}$+CLUSTER IS. In the following, we formally define the problem $\mathscr{F}$-DISJOINT CONFLICT FREE FEEDBACK VERTEX SET ($\mathscr{F}$-DCF-FVS, for short)

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

We note that in the definition of $\mathscr{F}$-DCF-FVS, there are three additional inputs (that is, W, R and C). The purpose and need for these sets will become clear when we describe the algorithm for $\mathscr{F}$-DCF-FVS. In Section 4.4.1, we will prove the following theorem.

Theorem 4.4.1. *Let $\mathscr{F}$ be a hereditary family of graphs for which there is an* FPT *algorithm for $\mathscr{F}$+*CLUSTER IS *running in time $f(k)n^{\mathcal{O}(1)}$, where n is the number of vertices in the input graph. Then, there is an* FPT *algorithm for $\mathscr{F}$-*DCF-FVS *running in time $16^k f(k)n^{\mathcal{O}(1)}$, where n is the (total) number of vertices in the input graphs.*

In the rest of the section, we show how we can use the FPT algorithm for $\mathscr{F}$-DCF-FVS

to obtain an FPT algorithm for $\mathscr{F}$-CF-FVS.

An Algorithm for $\mathscr{F}$-CF-FVS using the algorithm for $\mathscr{F}$-DCF-FVS: Let $I = (G,H,k)$ be an instance of $\mathscr{F}$-CF-FVS. We start by checking whether or not G has a feedback vertex set of size at most k, that is, a set Z of size at most k, such that $G-Z$ is a forest. For this we employ the algorithm for FEEDBACK VERTEX SET running in time $\mathcal{O}(3.619^k n^{\mathcal{O}(1)})$ of Kociumaka and Pilipczuk [70]. Here, n is the number of vertices in the input graph. Notice that if G does not have a feedback vertex set of size at most k, then (G,H,k) is a no-instance of $\mathscr{F}$-CF-FVS, and we can output a trivial no-instance of $\mathscr{F}$-DCF-FVS. Therefore, we assume that (G,k) is a yes-instance of FEEDBACK VERTEX SET, and let Z be one of its solution. We note that such a set Z can be computed using the algorithm presented in [70]. We generate an instance I_Y of $\mathscr{F}$-DCF-FVS, for each $Y \subseteq Z$, where Y is the guessed (exact) intersection of the set Z with an assumed (hypothetical) solution to $\mathscr{F}$-CF-FVS in I. We now formally describe the construction of I_Y. Consider a set $Y \subseteq Z$, such that Y is an independent set in H. Let $G_Y = G-Y$, $H_Y = H-Y$, $k_Y = k-|Y|$, $W_Y = Z \setminus Y$, $R_Y = (N_H(Y) \setminus W_Y) \cap V(H_Y)$, and $\mathscr{C}_Y = \emptyset$. Furthermore, let $I_Y = (G_Y, H_Y, k_Y, W_Y, R_Y, \mathscr{C}_Y)$, and notice that I_Y is a (valid) instance of $\mathscr{F}$-DCF-FVS. Now we resolve I_Y using the (assumed) FPT algorithm for $\mathscr{F}$-DCF-FVS, for each $Y \subseteq Z$, where Y is an independent set in H. It is easy to see that I is a yes-instance of $\mathscr{F}$-CF-FVS if and only if there is an independent set $Y \subseteq Z$ in H, such that I_Y is a yes-instance of $\mathscr{F}$-DCF-FVS. From the above discussions, we obtain the following lemma.

Lemma 19. *Let $\mathscr{F}$ be a family of graphs for which $\mathscr{F}$-DCF-FVS admits an FPT algorithm running in time $f(k)c^k n^{\mathcal{O}(1)}$, where n is the (total) number of vertices in the input graph. Then $\mathscr{F}$-CF-FVS admits an FPT algorithm running in time $f(k)(1+c)^k n^{\mathcal{O}(1)}$, where n is the number of vertices in the input graphs.*

Using Theorem 4.4.1 and Lemma 19, we obtain the main theorem of this section.

Theorem 4.4.2. *Let $\mathscr{F}$ be a hereditary family of graphs for which there is an FPT algorithm for $\mathscr{F}$+CLUSTER IS running in time $f(k)n^{\mathcal{O}(1)}$, where n is the number of*

vertices in the input graph. Then, there is an FPT *algorithm for* $\mathscr{F}$-CF-FVS *running in time* $17^k f(k)n^{\mathcal{O}(1)}$*, where n is the number of vertices in the input graphs of* $\mathscr{F}$-CF-FVS.

4.4.1 FPT Algorithm for $\mathscr{F}$-DCF-FVS

The goal of this section is to prove Theorem 4.4.1. Let $\mathscr{F}$ be a (fixed) hereditary family of graphs, for which $\mathscr{F}$+CLUSTER IS admits an FPT algorithm. We design a branching based FPT algorithm for $\mathscr{F}$-DCF-FVS, using the (assumed) FPT algorithm for $\mathscr{F}$+CLUSTER IS.

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

Reduction Rule 5. *Return that* $(G, H, k, W, R, \mathscr{C})$ *is a* no*-instance of* $\mathscr{F}$-DCF-FVS *if one of the following conditions are satisfied:*

1. $k < 0$,
2. $k = 0$ *and* G *has a cycle,*
3. $k = 0$ *and* $\mathscr{C} \neq \emptyset$,
4. $G[W]$ *has a cycle,*
5. $|\mathscr{C}| > k$, *or*
6. *there is* $C \in \mathscr{C}$, *such that* $C \subseteq R$.

Reduction Rule 6. *If* $k = 0$, G *is acyclic, and* $\mathscr{C} = \emptyset$, *then return that* $(G, H, k, W, R, \mathscr{C})$ *is a* yes*-instance of* $\mathscr{F}$-DCF-FVS.

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

Lemma 20. *Let $(G,H,k,W,R,\mathscr{C})$ be an instance of $\mathscr{F}$-*DCF-FVS*, where Reduction Rules 5 is not applicable and $G-W$ has no vertices. Then, in polynomial time, we can generate an instance (G',H',k') of $\mathscr{F}+$*CLUSTER IS*, such that $(G,H,k,W,R,\mathscr{C})$ is a* yes*-instance of $\mathscr{F}$-*DCF-FVS *if and only if (G',H',k') is a* yes*-instance of $\mathscr{F}+$*CLUSTER IS.

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

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

In the reverse direction, let (G',H',k') be a yes-instance of $\mathscr{F}+$CLUSTER IS, and S be one of its solution. Since Reduction Rule 5 (item 4) is not applicable on $(G,H,k,W,R,\mathscr{C})$, we have $|\mathscr{C}| \leq k$. Therefore, S is of size at most k. By non-applicability of item 6 of Reduction Rule 5, we have $S \cap R = \emptyset$. By construction, $|S \cap C| = 1$, for each $C \in \mathscr{C}$, and S is an independent set in H. From the above discussions, together with the fact that $G = G[W]$ is acyclic, implies that S is a solution to $\mathscr{F}$-DCF-FVS in $(G,H,k,W,R,\mathscr{C})$. This concludes the proof. □

Lemma 20 leads us to the following reduction rule.

Reduction Rule 7. *If $G-W$ has no vertices, then return the output of algorithm for $\mathscr{F}+$*CLUSTER IS *with the instance generated by Lemma 20.*

It is easy to see that Reduction Rules 5, 6, and 7 are safe.

Reduction Rule 8. *If there is a vertex $v \in V(G)$ of degree at most one in G, then return $(G-\{v\},H,k,W\setminus\{v\},R,\mathscr{C})$.*

The safeness of Reduction Rule 8 follows from the fact that a vertex of degree at most one does not participate in any cycle.

Reduction Rule 9. *Let $uv \in E(G)$ be an edge of multiplicity greater than 2 in G, and G' be the graph obtained from G by reducing the multiplicity of uv in G to 2. Then, return $(G', H, k, W, R, \mathscr{C})$.*

The safeness of Reduction Rule 9 follows from the fact that for an edge, multiplicity of 2 is enough to capture multiplicities of size larger than 2.

Reduction Rule 10. *Let $v \in R$ be a degree 2 vertex in G with u and w being its neighbors in G. Furthermore, let G' be the graph obtained from G by deleting v and adding the (multi) edge uw. Then, return $(G', H - \{v\}, k, W, R \setminus \{v\}, \mathscr{C})$.*

The safeness of Reduction Rule 10 follows from the fact that a vertex in R cannot be part of any solution and any cycle (in G) containing v must contain both u and w.

Reduction Rule 11. *If there is $v \in (V(G) \cap R)$, such that v has at least two neighbors in the same connected component of W, then return that $(G, H, k, W, R, \mathscr{C})$ is a* no*-instance of $\mathscr{F}$-*DCF-FVS.

Reduction Rule 12. *If there is $v \in V(G) \setminus (W \cup R)$, such that v has at least two neighbors in the same connected component of W, then return $(G - \{v\}, H - \{v\}, k - 1, W, R \cup N_H(v), \mathscr{C})$.*

Reduction Rule 13. *Let $v \in V(G) \cap R$, such that $N_G(v) \cap W \neq \emptyset$. Then, return $(G, H, k, W \cup \{v\}, R \setminus \{v\}, \mathscr{C})$.*

It is easy to see that Reduction Rules 11, 12, and 13 are safe.

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

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

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

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

Case 1. *If there is $v \in V(G) \setminus W$ that has at least two neighbors (in G), say $w_1, w_2 \in W$. Since Reduction Rule 11 and 12 are not applicable, w_1 and w_2 belong to different connected components of $G[W]$. Also, since Reduction Rule 13 is not applicable, we have $v \notin R$. In this case, we branch as follows.*

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

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

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

If Branching Rule 1 is not applicable, then each $v \in V(G) \setminus W$ has at most one neighbor

(in G) in the set W. Moreover, since Reduction Rule 8 is not applicable, each leaf in $G-W$ has a neighbor in W.

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

Proposition 11. *Let $(G,H,k,W,R,\mathscr{C})$ be an instance of $\mathscr{F}$-DCF-FVS, where none of Reduction Rule 5 to 13 or Branching Rule 1 apply. Then, there are vertices $u,v \in V(G)\setminus W$, such that the unique path P_{uv} in $G-W$ is a nice path.*

Consider $u,v \in V(G)\setminus W$, for which there is a nice path P_{uv} in T, where T is a connected component of $G-W$. Since Reduction Rule 8 is not applicable, either u has a neighbor in W, or u has degree at least 2 in T. From the above discussions, together with Proposition 11, we design the remaining branching rules used by the algorithm. We note that the branching rules that we describe next is similar to the one given in [4].

Case 2. *Let $v \in V(G)\setminus W$ be a leaf in $G-W$ for which the following holds. There is $u \in V(G)\setminus W$, such that $N_G(u)\cap W \neq \emptyset$ and there is a nice path P_{uv} from u to v in $G-W$. Let $C = V(P_{uv})\setminus\{u\}$, u' and v' be the neighbors (in G) of u and v in W, respectively. Observe that since Reduction Rule 13 is not applicable, we have $u,v \notin R$. We further consider the following cases, based on whether or not u' and v' are in the same connected component of $G[W]$.*

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

must contain all vertices in $V(P_{uv})$, consecutively. This leads to the following branching rule.

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

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

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

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

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

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

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

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

Case 3. *There is $u \in V(G) \setminus W$ which has (at least) two nice paths, say P_{uv_1} and P_{uv_2} to leaves v_1 and v_2 (in $G-W$). Let $C_1 = V(P_{uv_1}) \setminus \{u\}$ and $C_2 = V(P_{uv_2}) \setminus \{u\}$. We further*

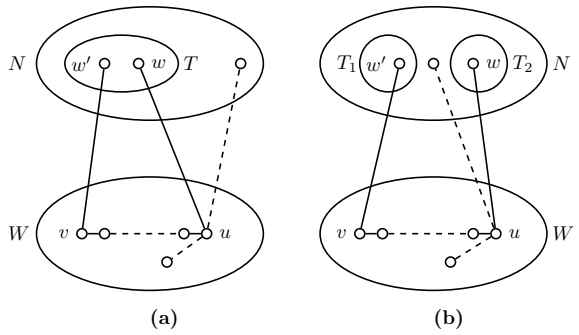

Figure 4.1: The cases handled by Branching Rule 2, (a) T is a connected component in $G[W]$, similarly in (b) T_1, T_2 are connected components in $G[W]$.

consider the following cases depending on whether or not v_1 and v_2 have neighbors (in G) in the same connected component of $G[W]$ and $u \in R$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

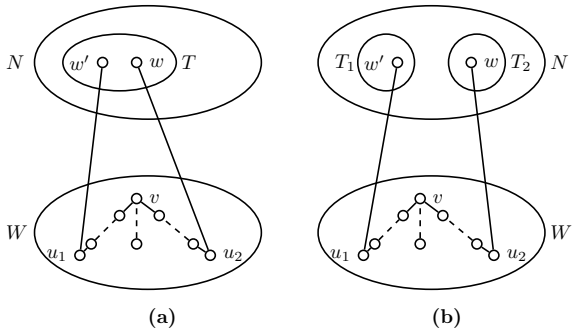

Figure 4.2: The cases handled by Branching Rule 3, In (a) T is a connected component in $G[W]$, similarly in (b) T_1, T_2 are connected components in $G[W]$.

This completes the description of the algorithm. We are now ready to prove Theorem 4.4.1.

Proof of Theorem 4.4.1. Let $I = (G, H, k, W, R, \mathscr{C})$ be an instance of $\mathscr{F}$-DCF-FVS, and n be the (total) number of vertices in G and H. We prove the correctness of our algorithm by induction on μ.

When $\mu \leq 0$, then Reduction Rule 5 or Reduction Rule 6, correctly resolve the given instance of $\mathscr{F}$-DCF-FVS. This forms the base case of our induction. For the induction hypothesis, we assume that for some $\delta \in \mathbb{N}$, for each $\mu \leq \delta$, the algorithm can correctly resolve the instance. The algorithm either applies one of Reduction Rule 5 to 13 or one of Branching Rule 1 to 3. Proposition 20 implies that either one of Reduction Rule 5 to 13

or Branching Rule 1 is applicable, or one of Branching Rule 2 to 3 is applicable. Each of the reduction rules are safe, they do not increase the measure, and can be applied only polynomially many times. Each of our branching rules are exhaustive, and in each of the branches, the measure strictly decreases. If we apply the reduction rules (exhaustively), either we completely resolve the instance correctly, or eventually apply a branching rule (in polynomial number of application of reduction rules). If one of the branching rules apply, then the measure strictly decreases, and then the induction hypothesis implies the correctness of the algorithm. This concludes the proof of correctness of the algorithm.

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

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

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

4.5 FPT Algorithm for $K_{i,j}$-free+CLUSTER IS

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

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

In Section 4.5.1, we design a polynomial time algorithm for the problem LARGE $K_{i,j}$-free+CLUSTER IS. In the rest of this section, we show how to use the polynomial time algorithm for LARGE $K_{i,j}$-free+CLUSTER IS to obtain an FPT algorithm for $K_{i,j}$-free+CLUSTER IS.

Theorem 4.5.1. *$K_{i,j}$-free+CLUSTER IS admits an FPT algorithm running in time $\mathcal{O}(k^{k^2} n^{\mathcal{O}(1)})$, where n is the number of vertices in the input graph.*

Proof. Let (G,H,k) be an instance of $K_{i,j}$-free+CLUSTER IS, and let $\mathscr{C} = \{C_1, C_2, \cdots, C_k\}$ be the set of connected components in H. If $k \leq 0$, we can correctly resolve the instance in polynomial time (by appropriately outputting yes or no answer). Therefore, we assume $k \geq 1$. If for each $C \in \mathscr{C}$, we have $|V(C)| \geq k^k$, then (G,H,k) is also an instance of LARGE $K_{i,j}$-free+CLUSTER IS, and therefore we resolve it in polynomial time using the algorithm for LARGE $K_{i,j}$-free+CLUSTER IS (Section 4.5.1). Otherwise, there is $C \in \mathscr{C}$, such that $|V(C)| < k^k$. Any solution to $K_{i,j}$-free+CLUSTER IS in (G,H,k) must contain exactly one vertex from C. Moreover, if a vertex $v \in V(C)$ is in the solution, then none of its neighbors in G and in H can belong to the solution. Therefore, we branch on vertices in C as follows. For each $v \in V(C)$, create an instance $I_v(G-(N_H(v) \cup N_G(v)), H-(N_H(v) \cup N_G(v)), k-1)$ of $K_{i,j}$-free+CLUSTER IS. If number of connected components in $H - N[C]$ is less than $k-1$, then we call such an instance I_v as *invalid* instance, otherwise the instance is a *valid* instance. Notice that for $v \in V(C)$, if I_v is an invalid instance, then v cannot belong to any solution. Thus, we branch on valid instances of I_v, for $v \in V(C)$. Observe that (G,H,k) is a yes-instance of $K_{i,j}$-free+CLUSTER IS if and only if there is a valid instance I_v, for

$v \in V(C)$, which is a yes-instance of $K_{i,j}$-free+CLUSTER IS. Therefore, we output the OR of results obtained by resolving valid instances I_v, for $v \in V(C)$.

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

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

4.5.1 Polynomial Time Algorithm for LARGE $K_{i,j}$-free+CLUSTER IS

Consider a (fixed) family of $K_{i,j}$-free graphs, where $1 \leq i \leq j$. The goal of this section is to design a polynomial time algorithm for LARGE $K_{i,j}$-free+CLUSTER IS. Let (G,H,k) be an instance of LARGE $K_{i,j}$-free+CLUSTER IS, where G is a $K_{i,j}$-free graph and H is a cluster graph with k connected components. We assume that $k > i + j + 2$, as otherwise, we

can resolve the instance in polynomial time (using brute-force). Let $\mathscr{C} = \{C_1, C_2, \cdots, C_k\}$ be the set of connected components in H, such that $|V(C_1)| \geq |V(C_2)| \geq \cdots \geq |V(C_k)|$.

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

Lemma 21. [16] *The number of edges in a $K_{i,j}$-free graph are bounded by $n^{2-\varepsilon}$, where $\varepsilon = \varepsilon(i,j) \in (0,1]$.*

Lemma 22. *Let (G,H,k) be an instance of* LARGE $K_{i,j}$*-free*+CLUSTER IS. *There exists $v \in V(C_1)$, such that for each $C \in \mathscr{C} \setminus \{C_1\}$, we have $|N_G(v) \cap C| \leq \frac{2j|C|}{k}$.*

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

Claim 1. $|E(C_1,C)| \leq jy^i + jx$.

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

$$\begin{aligned} |E(C_1,C)| &= \sum_{v \in C_1} |N_G(v) \cap V(C)| \\ &= \sum_{v \in C_h^1} |N_G(v) \cap V(C)| + \sum_{v \in C_l^1} |N_G(v) \cap V(C)|. \end{aligned}$$

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

Let $A_{\mathsf{deg}}(C_1,C)$ denote average degree of vertices in set C_1 in $G[E(C_1,C)]$. That is, $A_{\mathsf{deg}}(C_1,C) = \frac{|E(C_1,C)|}{|C_1|}$. In the following claim, we give a bound on $A_{\mathsf{deg}}(C_1,C)$.

Claim 2. $A_{\mathsf{deg}}(C_1, C) \leq \frac{2jy}{k^2}$.

Proof. From Claim 1, we have $|E(C_1, C)| \leq jy^i + jx$. Therefore, $A_{\mathsf{deg}}(C_1, C) \leq j + \frac{jy^i}{x}$. Using Lemma 21, we have $A_{\mathsf{deg}}(C_1, C) \leq \frac{(x+y)^{2-\varepsilon}}{x} \leq 4x^{1-\varepsilon}$. To prove the claim, us consider the following cases:

Case 1. $x \geq k^2 y^{i-1}$. In this case, using the inequality $A_{\mathsf{deg}}(C_1, C) \leq j + \frac{jy^i}{x}$, we have $A_{\mathsf{deg}}(C_1, C) \leq j + \frac{jy}{k^2}$. Since $y > k^2$ (and $k > 5$), we have $A_{\mathsf{deg}}(C_1, C) \leq \frac{2jy}{k^2}$.

Case 2. $x < k^2 y^{i-1}$. In this case, we use the inequality $A_{\mathsf{deg}}(C_1, C) \leq 4x^{1-\varepsilon}$, to obtain $A_{\mathsf{deg}}(C_1, C) < 4k^{2(1-\varepsilon)} y^{(i-1)(1-\varepsilon)} < \frac{4k^2 y}{y^{(2-i)+\varepsilon(i-1)}}$. Since $y \geq k^k$, we have $y^{(2-i)+\varepsilon(i-1)} > \frac{2k^4}{j}$. Therefore, we have $A_{\mathsf{deg}}(C_1, C) < \frac{2jy}{k^2}$. □

In the following, we will give a probabilistic argument on the existence of a vertex with the desired properties in the lemma statement. For $v \in V(C_1)$, let $\mathsf{deg}(v, C)$ denote the size of $|N_G(v) \cap V(C)|$. From Claim 2, we have $A_{\mathsf{deg}}(C_1, C) \leq \frac{2jy}{k^2}$. Using Markov's inequality, the upper bound on the probability that $\mathsf{deg}(v, C) \geq \frac{2jy}{k}$ is $P(\mathsf{deg}(v, C) \geq \frac{2jy}{k}) \leq \frac{1}{k}$. Using Boole's inequality (the union bound), the probability that $\mathsf{deg}(v, C)$ is greater than or equal to $\frac{2j|C|}{k}$ for at least one $C \in \mathscr{C} \setminus \{C_1\}$ is bounded by $P(\cup_{C \in \mathscr{C} \setminus \{C_1\}} \mathsf{deg}(v, C) \geq \frac{2j|C|}{k}) \leq \frac{1}{k}.(k-1) < 1$. This implies that probability that $\mathsf{deg}(v, C) \leq \frac{2j|C|}{k}$, for each $C \in \mathscr{C} \setminus \{C_1\}$ is greater than 0. This completes the proof. □

We are now ready to describe our algorithm, which is given in Algorithm 1.

Lemma 23. *Algorithm 1 for* LARGE $K_{i,j}$*-free+*CLUSTER IS *is correct and runs in polynomial time.*

Proof. We first prove the correctness of the algorithm using induction on, t. The base case is when $1 \leq t \leq 2j$. The algorithm correctly resolve the instance using brute force. For the induction hypothesis, we assume that the algorithm is correct for each $t \leq d-1$. Next, we show that the algorithm is correct for $t = d$. Let $C_1, \cdots, C_d$ be the set of connected

Algorithm 1 (G,H,k) : Greedy algorithm for LARGE $K_{i,j}$-free+CLUSTER IS

1: $t = k$ and $S = \emptyset$;
2: **while** $t > 2j$ **do**
3: Let $C_1, \cdots, C_t$ be the connected components of H, sorted in decreasing order of their sizes;
4: Let $v \in V(C_1)$ be a vertex which satisfies the condition of Lemma 22;
5: Add v to S;
6: Decrease t by 1;
7: $G = G - (N_G(v) \cup N_H[v])$ and $H = H - (N_G(v) \cup N_H[v])$;
8: **end while**
9: Solve (G,H,t) by a brute force algorithm, as $t \leq 2j$;

components in H, sorted in decreasing order of their sizes. By Lemma 22, there is $v \in C_1$, such that for each $C \in \mathscr{C} \setminus \{C_1\}$, we have $\deg(v,C) \leq \frac{2j|C|}{d}$.

We delete all vertices in $N_H[v] \cup N_G(v)$ from G and H. Observe that from each $C \in \mathscr{C} \setminus \{C_1\}$, we have deleted at most $\frac{2j|C|}{d}$ vertices, which are neighbors of v in G. Let $C' = C \setminus (N_H[v] \cup N_G(v)) = C \setminus N_G(v)$. It is enough to show that $|C'| \geq (d-1)^{(d-1)}$. Note that $|C'| \geq |C| - \frac{2j|C|}{d}$. As base case is not applicable, we can assume that $d > 2j$. Hence, $|C'| \geq |C|(1 - \frac{2j}{d}) \geq d^d(1 - \frac{2j}{d}) \geq d^{d-1}(d-2j) \geq (d-1)^{(d-1)}$.

This concludes the proof of correctness of the algorithm. At each step we either sort the components on the basis of their size or find a vertex of lower degree which can be carried out in polynomial time, or solve the instance using brute force approach, where the solution size we are seeking for is bounded by a constant (at most $2j$). Moreover, the algorithm terminates after at most k iterations. Thus, the running time of the algorithm is bounded by a polynomial in the size of the input. □

Using Lemma 23, we obtain the following theorem.

Theorem 4.5.2. *The problem* LARGE $K_{i,j}$*-free*+CLUSTER IS *admits a polynomial time algorithm.*

4.6 Conclusion

In this chapter we studied the $\mathscr{F}$-CF-FVS problem, which is a conflict-free variant of FVS where H belongs the class $\mathscr{F}$. We obtained a complete dichotomy result on the Parameterized Complexity of the problem $\mathscr{F}$-CF-FVS, when $\mathscr{F}$ is a hereditary graph family by relating $\mathscr{F}$-CF-FVS to the INDEPENDENT SET problem on special classes of graphs. In particular, we showed that $\mathscr{F}$-CF-FVS is FPT parameterized by the solution size if and only if $\mathscr{F}+$CLUSTER IS is FPT parameterized by the solution size. Here, $\mathscr{F}+$CLUSTER IS is the INDEPENDENT SET problem in the (edge) union of a graph $G \in \mathscr{F}$ and a cluster graph H (G and H are explicitly given). Next, we exploited this characterization to obtain new FPT results as well as intractability results for $\mathscr{F}$-CF-FVS. We gave an FPT algorithm for $\mathscr{F}+$CLUSTER IS when $\mathscr{F}$ is the family of $K_{i,j}$-free graphs. We showed that for the family of bipartite graph $\mathscr{B}$, $\mathscr{B}$-CF-FVS is W[1]-hard, when parameterized by the solution size. Finally, we considered, for each $0 < \varepsilon < 1$, the family of graphs $\mathscr{F}_\varepsilon$, which comprise of graphs G such that $|E(G)| \leq |V(G)|^{2-\varepsilon}$, and showed that $\mathscr{F}_\varepsilon$-CF-FVS is W[1]-hard, when parameterized by the solution size, for every $0 < \varepsilon < 1$.

Chapter 5

Exploring the Kernelization Borders for Hitting Cycles

5.1 Introduction

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 [13, 43] 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. [102] 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. [18] resolved the question in the affirmative by designing a kernel of size $\mathcal{O}(k^{11})$. Later, Bodlaender [14] reduced the kernel size to $\mathcal{O}(k^3)$, and finally Thomassé [105] designed a kernel of size $\mathcal{O}(k^2)$. The kernel of Thomassé [105] is best possible under a well known complexity theory hypothesis. It is important to emphasize that [105] 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 [71] 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. [92] 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 chapter we study the kernelization complexity of the conflict-free generalization of FVS, and OCT, which we defined in previous chapters. We recall the problems here.

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?

One can similarly define CONFLICT FREE ODD CYCLE TRANSVERSAL (CF-OCT).

In previous chapters we observed that, if one wants to model "independent" version of

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

these problems (where the solution is suppose to be an independent set), then one takes conflict graph to be same as the input graph. We also observed that these problems are W[1]-hard on general graphs, a simple reduction from MULTICOLOR INDEPENDENT SET with each color class being modeled as cycle and the conflict graph being the input graph. Next, we studied the question that: *when does the problem become* FPT? To state the question formally, we let $\mathscr{G}$ and $\mathscr{H}$ be two families of graphs. Then, $(\mathscr{G}, \mathscr{H})$-CF-FVS is same problem as CF-FVS, but the input graph G and the conflict graph H are restricted to belong to $\mathscr{G}$ and $\mathscr{H}$, 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 lead to some fruitful and rewarding research.

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 $\mathscr{D}_d$ denote the set of graphs of *degeneracy* at most d. In this chapter we study the $(\star, \mathscr{D}_d)$-CF-FVS ($\mathscr{D}_d$-CF-FVS) problem. The symbol $\star$ denotes that the input graph G is arbitrary. One can similarly define $\mathscr{D}_d$-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 $\mathscr{P}^{\star\star}_{\leq 3}$ and this variant of CF-OCT as $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT.

In the previos chapters we studied conflict-free graph modification problems in the realm of Parameterized Complexity. We also gave FPT algorithms for $\mathscr{D}_d$-CF-FVS, $\mathscr{D}_d$-CF-OCT and $\mathscr{D}_d$-CF-ECT using the independence covering families [81]. We also gave FPT algorithms for these problems when the conflict graph belongs to nowhere dense graphs. In this chapter we focus on the kernelization complexity of $\mathscr{D}_d$-CF-FVS, and $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT obtain the following results.

1. $\mathscr{D}_d$-CF-FVS admits a $\mathscr{O}(k^{\mathscr{O}(d)})$ kernel.

2. $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT does not admit polynomial kernel, unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Note that $\mathscr{D}_0$ denotes edgeless graphs and hence $\mathscr{D}_0$-CF-FVS, and $\mathscr{D}_0$-CF-OCT are essentially FVS, and OCT, respectively. Thus, any polynomial kernel for $\mathscr{D}_d$-CF-FVS, and $\mathscr{P}^{\star\star}_{\leq 3}$-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 $\mathscr{O}(k^3)$. 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 $\mathscr{O}(k^3)$. That is, $G' - V_{\geq 3}$ consists of $\mathscr{O}(k^3)$ connected components where each component is a path.

We obtain this structural decomposition using reduction rules inspired by the quadratic kernel for FVS [105]. 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 cannot do this for any arbitrary graph H as the problem is W[1]-hard. Once the decomposition is obtained we cannot 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 $\mathscr{D}_d$-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 [15]. We first define a conflict version of the s-t-CUT problem, where H belongs to $\mathscr{P}^{\star\star}_{\leq 3}$. 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 $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT, and obtain the desired kernel lower bound.

5.2 Preliminaries

Throughout the chapter, 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_1v_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 [87].

5.3 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 $\mathscr{D}_d$-CF-FVS (Section 5.4). 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 $\mathscr{D}_d$-CF-FVS.

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

Definition 12. 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 13. 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 [87]. 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 $N_H^f(v)$ ($N_H^b(v)$) 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 14. 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 $Z \subseteq N_H^f(u)$, it holds that $|U| \leq n_q$.

Now, we give our polynomial time algorithm for d-BIP in Algorithm 2.

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

Observation 5.3.1. *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 $u \in U$, $N_H^b(u) \cap S \neq \emptyset$, we have that $|U| \leq kd$.*

Now we are ready to prove the correctness of our algorithm (Algorithm 2) for d-BIP.

Lemma 24. *Algorithm 2 is correct.*

Algorithm 2 Algo1(H,X)

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

1: For $q \in [d]$, set $n_q = kd+1$, when $q = 1$, and $n_q = kn_{q-1} + kd + k + 1$, otherwise.
2: $q = 1$.
3: **while** $q \leq d$ **do**
4: **while** X is not q-reducible **do**
5: Find $U \subseteq X$ of size $n_q + 1$, for which there is $Z \subseteq V(H)$ of size exactly $d - q + 1$, such that for each $u \in U$, we have $Z \subseteq N_H^f(u)$.
6: Let v be an arbitrary vertex in U.
7: $X = X \setminus \{v\}$.
8: **end while**
9: $q = q + 1$.
10: **end while**
11: **while** $|X| > n_{d+1}$ **do**
12: Let v be an arbitrary vertex in X.
13: $X = X \setminus \{v\}$.
14: **end while**
15: Set $X' = X$.
16: **return** X'

Proof. Let (H,X,k) be an instance of d-BIP, and X' be the output returned by Algorithm 2 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 $(\cup_{s \in S} N_H^f(s)) \cap X'$. 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 5.3.1). 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 U_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 $q^* \leq q'$, 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 25. *Algorithm 2 runs in time $n^{\mathcal{O}(d)}$.*

Proof. Let (H, X, k) be an instance of d-BIP, and σ be a (fixed) d-degeneracy sequence of H (which can computed in polynomial time [87]). Using σ, for each $v \in X$, we can find $N_H^f(v)$ in the polynomial time. For each set of size at most $d+1$, we can find all the vertices in $V(H)$ that have all of them in their neighborhood in polynomial time. Thus, Step 4 of the algorithm can be executed in polynomial time (for fixed d). Also, all the other steps of the algorithm can be performed in time $n^{\mathcal{O}(d)}$ time. This completes the proof. □

□

Using Lemma 24 and Lemma 25 we obtain the following theorem.

Theorem 5.3.2. *d-BOUNDED INDEPENDENCE PRESERVER admits an algorithm running in time $n^{\mathcal{O}(d)}$.*

5.4 A Polynomial Kernel for $\mathscr{D}_d$-CF-FVS

In this section, we design a kernelization algorithm for $\mathscr{D}_d$-CF-FVS. To design a kernelization algorithm for $\mathscr{D}_d$-CF-FVS, we define another problem called $\mathscr{F}$-DISJOINT CONFLICT FREE FEEDBACK VERTEX SET ($\mathscr{F}$-DCF-FVS, for short). We first define the problem $\mathscr{F}$-DCF-FVS formally, and then explain its uses in our kernelization algorithm.

$\mathscr{F}$-DISJOINT CONFLICT FREE FEEDBACK VERTEX SET ($\mathscr{F}$-DCF-FVS)

Input: An undirected graph G, a graph $H \in \mathscr{D}_d$ such that $V(G) = V(H)$, a subset $R \subseteq V(G)$, and a non-negative integer k.

Parameter: 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 $\mathscr{D}_d$-CF-FVS is a special case of $\mathscr{F}$-DCF-FVS, where $R = \emptyset$. Given an instance of $\mathscr{D}_d$-CF-FVS, the kernelization algorithm creates an instance of $\mathscr{F}$-DCF-FVS by setting $R = \emptyset$. Then it applies a kernelization algorithm for $\mathscr{F}$-DCF-FVS. Finally, the algorithm takes the instance returned by the kernelization algorithm for $\mathscr{F}$-DCF-FVS and generates an instance of $\mathscr{D}_d$-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 $\mathscr{F}$-DCF-FVS, and then give a polynomial time linear parameter preserving reduction from $\mathscr{F}$-DCF-FVS to $\mathscr{D}_d$-CF-FVS. If the kernelization algorithm for $\mathscr{F}$-DCF-FVS returns that (G,H,R,k) is a yes (no) instance of $\mathscr{F}$-DCF-FVS, then conclude that (G,H,k) is a yes (no) instance of $\mathscr{D}_d$-CF-FVS.In the following, we describe a kernelization algorithm for $\mathscr{F}$-DCF-FVS. Let (G,H,R,k) be an instance of $\mathscr{F}$-DCF-FVS. The algorithm starts by applying the following simple reduction rules.

Reduction Rule 14.

(a) If $k \geq 0$ and G is acyclic, then return that (G,H,R,k) is a yes*-instance of $\mathscr{F}$-*DCF-FVS.

(b) Return that (G,H,R,k) is a no*-instance of $\mathscr{F}$-*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*

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

Reduction Rule 15.

(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* $\mathscr{F}$-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* $\mathscr{F}$-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 15. 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 12. [29, 90, 105] *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 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 $\mathscr{C}_v$ of components of $G[V(G) \setminus (X_v \cup \{v\})]$, such that each component in $\mathscr{C}_v$ is a tree, v has exactly one neighbor in $C \in \mathscr{C}_v$, and there exist at least $k+2$ components in $\mathscr{C}_v$ 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 16. *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 $\mathscr{F}$-*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 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 12 (item (i)).

Reduction Rule 17. *Let $v \in V(G)$, $X_v \subseteq V(G) \setminus \{v\}$, and $\mathscr{C}_v$ be the set of components which satisfy the conditions in Proposition 12(ii) (in G), then delete edges between v and the components of the set $\mathscr{C}_v$, and add parallel edges between v and every vertex $x \in X_v$ in G.*

The polynomial time applicability of Reduction Rule 17 follows from Proposition 12. And, in the following lemma, we prove the safeness of this reduction rule.

Lemma 26. *Reduction Rule 17 is safe.*

Proof. Let (G,H,R,k) be an instance of $\mathscr{F}$-DCF-FVS. Furthermore, let $v \in V(G)$, $X_v \subseteq V(G)$, and $\mathscr{C}_v$ be the tuple for which the conditions of Reduction Rule 17 are satisfied, and

(G',H,R,k) be the instance resulting after application of the reduction rule. We prove that (G,H,R,k) is a yes-instance of $\mathscr{F}$-DCF-FVS if and only if (G',H,R,k) is a yes-instance of $\mathscr{F}$-DCF-FVS.

In the forward direction, let (G,H,R,k) be a yes-instance of $\mathscr{F}$-DCF-FVS and S be one of its solution. We claim that S is also a solution of $\mathscr{F}$-DCF-FVS for (G',H,R,k). Suppose not, then $G'-S$ must contains a cycle as the conflict graphs in both the instances are the same. Observe that $G-\{v\}$ is identical to $G'-\{v\}$, and $G'-X_v$ is a subgraph of $G-X_v$, therefore, if either $v \in S$ or $X_v \subseteq S$, then S is a solution of $\mathscr{F}$-DCF-FVS for (G',H,R,k). Next, we assume that neither $v \notin S$, nor $X_v \nsubseteq S$. For $x \in X$, let $\mathscr{W}_x \subseteq \mathscr{C}_v$ be the set of components associated with x, which is obtained by the algorithm in Proposition 12(ii). Observe that, there are at least $k+2$ disjoint paths from v to each $x \in X_v$ passing through components in $\mathscr{W}_x$ in the graph G. Since S is of size at most k, there are at least two (distinct) connected components say C^1, C^2 in $\mathscr{W}_x$ such that v,x together with C^1, C^2 creates a cycle in $G-S$. This is a contradiction to S being a solution of $\mathscr{F}$-DCF-FVS for (G,H,R,k).

In the reverse direction, let (G',H,R,k) be a yes-instance of $\mathscr{F}$-DCF-FVS and S' be one of its solution. Observe that for each vertex $x \in X_v$, we have parallel edges between v and x in G', therefore either $v \in S'$ or $X_v \subseteq S'$. As observed before $G-\{v\}$ is identical to $G'-\{v\}$, therefore if $v \in S'$ then S' is also a solution of $\mathscr{F}$-DCF-FVS in (G,H,R,k). Next we assume that $X_v \subseteq S'$. Observe that edges incident to v and a vertex in some components in $\mathscr{C}_v$ are cut edges in $G-X_v$, by Proposition 12(ii), and hence they do not participate in any cycle in $G-X_v$. This concludes that S' is a solution of $\mathscr{F}$-DCF-FVS for (G,H,R,k). □ □

In the following state an easy observation, which follows from non-applicability of Reduction Rule 14 to 17.

Observation 5.4.1. *Let (G,H,R,k) be an instance of $\mathscr{F}$-DCF-FVS, where none of Reduction Rule 14 to 17 apply. Then the degree of each vertex in G is bounded by $\mathcal{O}(k^2)$.*

To design our next reduction rule, we construct an auxilary 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 refereed to as high degree vertices. For high degree vertex $v \in G$. 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 13. [29] *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 18. *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 27. *Let (G, H, R, k) be an instance of $\mathscr{F}$-*DCF-FVS*, where none of the Reduction Rules 14 to 18 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 19. *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 cannot 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 reduced 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 5.3. This immediately gives us the following reduction rule.

Reduction Rule 20. *Let P be a maximal degree two induced path in G. If $|V(P)| \geq n_{d+1}+1$, apply Algorithm 2 with input $(H, V(P) \setminus R)$. Let $\widehat{V}(P)$ be the set returned by Algorithm 2. 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 28. *Reduction Rule 20 is safe.*

Proof. Let (G, H, R, k) be an instance of $\mathscr{F}$-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 28 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 $\mathscr{F}$-DCF-FVS if and only if (G', H', R, k) is a yes-instance of $\mathscr{F}$-DCF-FVS.

In the forward direction, let (G, H, R, k) be a yes-instance of $\mathscr{F}$-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 $\mathscr{F}$-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. Bu then $(C \setminus \{xy\}) \cup \{xv, vy\}$ is a cycle in $G - S$. Next, we consider the case when $v \in S$. By Lemma 24 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 $\mathscr{F}$-DCF-FVS for (G', H', R, k).

In the reverse direction, let (G', H', R, k) be a yes-instance of $\mathscr{F}$-DCF-FVS and S' be one of its minimal solution. We claim that S' is also a solution of $\mathscr{F}$-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 contians 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 $\mathscr{F}$-DCF-FVS for (G, H, R, k). □

Lemma 29. *Let (G, H, R, k) be an instance of $\mathscr{F}$-DCF-FVS, where none of the Reduction Rules 14 to 20 are applicable. Then, the number of vertices in a degree two induced path in G is bounded by $\mathscr{O}(k^{\mathscr{O}(d)})$.*

Theorem 5.4.2. *$\mathscr{F}$-DCF-FVS admits a kernel with $\mathscr{O}(k^{\mathscr{O}(d)})$ vertices.*

Proof. Let (G, H, R, k) be an instance of $\mathscr{F}$-DCF-FVS, where none of the Reduction Rules 14 to 20 are applicable. Then, by Lemma 27, the number of vertices of degree at least 3 and the number of maximal degree two induced paths in G are bounded by $\mathscr{O}(k^3)$ and By Lemma 29, the number of vertices in a degree two induced path in G is bounded by $\mathscr{O}(k^{\mathscr{O}(d)})$. Hence, the number of vertices in G is bounded by $\mathscr{O}(k^{\mathscr{O}(d)})$. Since, each of the reduction rules can be applied in polynomial time and each of them either (correctly) declare that the given instance is a yes or no-instance or (safely) reduce the size of G. Therefore, the overall running time is polynomial in the input size. □

Lemma 30. *There is a polynomial time parameter preserving reduction from $\mathscr{F}$-DCF-FVS to $\mathscr{D}_d$-CF-FVS.*

Proof. Given an instance (G,H,R,k) of $\mathscr{F}$-DCF-FVS, we generate an instance (G',H',k') of $\mathscr{D}_d$-CF-FVS as follows. We let the vertex set of $V(G')$ and $V(H')$ to be $V(G)\cup\{x\}$, where x is a new vertex. Now, we define the edge sets of G' and H'. Initially, $E(G')=E(G)$. Additionally, we add a self loop on x in G'. We let $E(H')=E(H-R)\cup\{xw \mid w\in R\}$. We set $k'=k+1$. Clearly, this construction can be carried out in the running time linear in the size of the input instance. We claim that (G,H,R,k) is a yes-instance of $\mathscr{F}$-DCF-FVS if and only if $(G',H',k+1)$ is a yes-instance of $\mathscr{D}_d$-CF-FVS.

In the forward direction, let S be a solution to $\mathscr{F}$-DCF-FVS in (G,H,R,k). We claim that $S'=S\cup\{x\}$ is a solution to $\mathscr{D}_d$-CF-FVS in $(G',H',k+1)$. Since, $G'-\{x\}$ is identical to G, $G'-S'$ does not contain any cycle. Since, $S\cap R=\emptyset$, $S\cup\{v\}$ is an independent set in H'. This completes the proof in the forward direction. In the reverse direction, let (G',H',k') be a yes-instance of $\mathscr{D}_d$-CF-FVS and S be one of its solution. Since there is a self loop at x in G, $x\in S$. We claim that $S'=S\setminus\{x\}$ is a solution to $\mathscr{F}$-DCF-FVS in (G,H,R,k). Clearly, $G'-\{x\}$ is identical to G, therefore, $G-S'$ does not contain any cycle. Since, $x\in S$, none of the vertices in R can belong to S. Since, $H-R$ same as $H-(R\cup\{x\})$, S' is an independent set in H' and $S'\cap R=\emptyset$, we have that S' is a solution to $\mathscr{F}$-DCF-FVS in (G,H,R,k). □

By Theorem 5.4.2 and Lemma 30, we obtain the following result.

Theorem 5.4.3. $\mathscr{D}_d$-CF-FVS *admits a kernel with* $\mathscr{O}(k^{\mathscr{O}(d)})$ *vertices.*

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

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

$\mathscr{P}^{\star}_{\leq 3}$-CONFLICT FREE s-t CUT ($\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT) to $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT. $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT is formally defined as follows.

$\mathscr{P}^{\star}_{\leq 3}$-CONFLICT FREE s-t CUT ($\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT) **Parameter:** k
Input: An undirected graph G, a graph $H \in \mathscr{P}_{\leq 3}$ ($V(G) = V(H)$), two vertices s and t and an integer k
Question: Is there a set $X \subseteq V$ such that X is a $s-t$ cut in G and $H[X]$ is edgeless?

We first prove that $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT is NP-hard. Then, we prove that $\mathscr{P}^{\star}_{\leq 3}$-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. To show the NP-hardness of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT, we give a reduction from the well known NP-hard problem $(3,B2)$-SAT [11] to $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT. $(3,B2)$-SAT is formally defined as follows.

$(3,B2)$-SAT
Input: An instance $(U,\mathscr{C})$, where U is the set of boolean variables and $\mathscr{C}$ is the set of clauses such that each clause has exactly three literals, and each literal occurs in exactly two clauses
Question: Does there exist an assignment to variables such that each clause is satisfied?

5.5.1 NP-hardness of $\mathscr{P}_{\leq 3}$-CF-s-t CUT

In this section, we prove that $\mathscr{P}_{\leq 3}$-CF-s-t CUT is NP-hard. Given an instance $(U,\mathscr{C})$ of $(3,B_2)$-SAT, we construct an instance (G,H,s,t,k) of $\mathscr{P}_{\leq 3}$-CF-s-t CUT as follows. Let $|U| = n$ and $|\mathscr{C}| = m$. For each clause $C = (v_1,v_2,v_3) \in \mathscr{C}$, add vertices v_1^C, v_2^C, and v_3^C in $V(G)$ and $V(H)$. We also add $2n+2$ new vertices s, t, a_i and b_i in $V(G)$ and $V(H)$, where $i \in [n]$. Corresponding to each clause $C = (v_1,v_2,v_3) \in \mathscr{C}$, we add a path (s,v_1^C,v_2^C,v_3^C,t) in G. We also add paths (s,a_i,b_i,t), for all $i \in [n]$. Now we define edge set of H. Let $x_i \in U$. Add edges between a_i and vertices corresponding to positive literal of x_i and also between b_i and vertices corresponding to negative literal of x_i. We set $k = n+m$. Figure

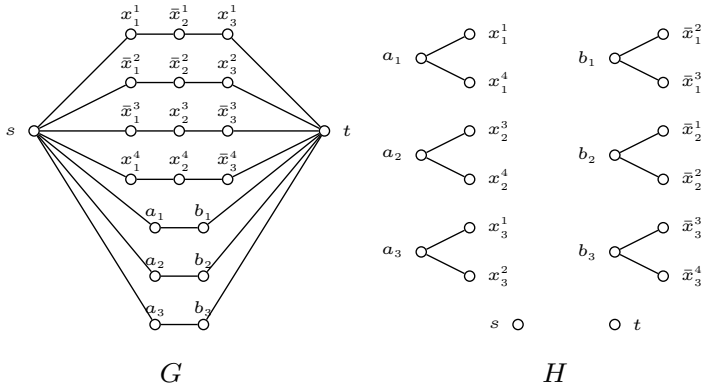

Figure 5.1: An illustration of construction of graph G and H in NP-hardness of $\mathscr{P}_{\leq 3}$-CF-s-t CUT for $\mathscr{C} = \{(x_1, \bar{x}_2, x_2), (\bar{x}_1, \bar{x}2, x_3), (\bar{x}_1, x_2, \bar{x}_3), (x_1, x_2, \bar{x}_3)\}$.

5.1 describes the construction of G and H. Clearly, this construction can be carried out in the polynomial time. In the following lemma, we prove that $\mathscr{C}$ is satisfiable if and only if (G,H) has a conflict-free $s-t$ cut of size $n+m$.

Lemma 31. *$(U,\mathscr{C})$ is a* yes*-instance of $(3,B_2)$-*SAT *if and only if (G,H,s,t,k) is a* yes*-instance of $\mathscr{P}_{\leq 3}$-*CF-s-t CUT.

Proof. In the forward direction, let $\mathscr{C}$ be satisfiable, and ϕ be a solution. Further, let S be the set of literals which are set to *true* in ϕ. Given S, we construct a solution S' of $\mathscr{P}_{\leq 3}$-CF-s-t CUT in (G,H) as follows. Let $v_i \in S$ and v_i belongs to the clauses C and C'. Add v_i^C and $v_i^{C'}$ to S'. Let $P_C = (s, v_1^C, v_2^C, v_3^C, t)$ be a path in G corresponding to the clause C. If more than one vertex from P_C belongs to S', delete all but one from S' arbitrarily. If variable corresponding to positive literal x_i belongs to S', add b_i to S', otherwise add a_i to S'. Since, there are $n+m$ disjoint paths between s and t and we select exactly one vertex from each path, $|S'| = n+m$. Since, $\mathscr{C}$ is satisfiable and for each path (s, a_i, b_i, t) either a_i or b_i belongs to S', S' is a $s-t$ cut of G. By the construction of S', it is also an independent set in H. This completes the proof in the forward direction.

In the reverse direction, let S be a solution to $\mathscr{P}_{\leq 3}$-CF-s-t CUT in (G,H,s,t,k). Given S, we construct a satisfyning assignment ϕ for the instance $(U,\mathscr{C})$ of $(3,B2)$-SAT as

follows. Let v be a literal which occurs in the clauses C and C'. If $S \cap \{v^C, v^{C'}\} \neq \emptyset$, we assign 1 to v. Since, $H[S]$ is edgeless, if vertex corresponding to positive literal x_i belongs to the solution, b_i belongs to the solution and hence vertices corresponding to negative literal $\bar{x}_i$ do not belong to the solution. This implies that both the positive and negative literal corresponding to a variable are not set to one. If none of them are true, we assign 1 to x_i (or to $\bar{x}_i$). By the construction of G, ϕ is a satisfying assignment for $\mathscr{C}$. □

Theorem 5.5.1. *$\mathscr{P}_{\leq 3}$-CF-s-t CUT is NP-hard.*

Proof. The proof follows from the construction of an instance of $\mathscr{P}_{\leq 3}$-CF-s-t CUT, Lemma 31 and NP-hardness of $(3, B_2)$-SAT. □

5.5.2 Lower bound for Kernel of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT

In this section, we prove that $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT does not admit a polynomial compression unless NP $\subseteq \frac{\text{coNP}}{\text{poly}}$ which results into the fact that $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT does not admit polynomial kernel as well. Towards this, we cross-compose $\mathscr{P}_{\leq 3}$-CF-s-t CUT into $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT parameterized by k, the size of cut. Before going into the details, we define the notion of cross-composition.

Definition 16. [15, 29] Let Σ be a finite set of alphabets. A *polynomial equivalence relation* is an equivalence relation $\mathscr{R}$ on $\Sigma^{\star}$ if there is an algorithm that given two strings $x, y \in \Sigma^{\star}$, decides whether $x \equiv_{\mathscr{R}} y$ in time polynomial in $|x| + |y|$. Moreover, the relation $\mathscr{R}$ restricted to the set $\Sigma^{\leq n}$ has at most $p(n)$ equivalence classes, where $p(\cdot)$ is some polynomial function.

Definition 17. [15, 29] Let $L \subseteq \Sigma^{\star}$ be a language and $Q \subseteq \Sigma^{\star} \times \mathbb{N}$ be a parameterized language. We say that L *cross-composes* into Q if there exists a polynomial equivalence relation $\mathscr{R}$ and an algorithm $\mathscr{A}$ satisfying the following conditions. The algorithm $\mathscr{A}$ takes as input a sequence of strings $x_1, \cdots, x_t \in \Sigma^{\star}$ that are equivalent with respect to $\mathscr{R}$, runs in time polynomial in $\Sigma_{i=1}^{t} |x_i|$, and outputs one instance $(y, k) \in \Sigma^{\star} \times \mathbb{N}$ such that:

(i) $k \leq p(\max_{i\in[t]} |x_i| + \log t)$ for some polynomial $p(\cdot)$, and

(ii) $(y,k) \in Q$ if and only if there exists at least one index $i \in [t]$ such that $x_i \in L$.

Now, we state following known result which will be further used in this section.

Theorem 5.5.2. [15, 29] *Let L be an NP-hard language that cross-composes into a parameterized language Q. Then, Q does not admit a polynomial compression, unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Next, we present a cross-composition of $\mathscr{P}_{\leq 3}$-CF-*s*-*t* CUT into $\mathscr{P}^{\star}_{\leq 3}$-CF-*s*-*t* CUT parameterized by the solution size.

Lemma 32. *There exists a cross-composition from* $\mathscr{P}_{\leq 3}$-CF-*s*-*t* CUT *into* $\mathscr{P}^{\star}_{\leq 3}$-CF-*s*-*t* CUT *parameterized by the cut size.*

Proof. By choosing an appropriate polynomial equivalence relation $\mathscr{R}$, we may assume that we are given q instances $(G_i, H_i, s_i, t_i, k_i)_{i=1}^{q}$ of $\mathscr{P}_{\leq 3}$-CF-*s*-*t* CUT, where $V(G_i) = n$ and k_i is same for each $i \in [q]$. More precisely, equivalence relation $\mathscr{R}$ is defined as follows. We put all malformed instances into one equivalent class, while all the well-formed instances are partitioned with respect to the number of vertices in the graph and the integer k_i, where $i \in [q]$. Two well-formed instances are considered equivalent if number of the vertices in the graphs and integer k_i are same in both the instances. Clearly, the number of equivalence relation in $\Sigma^{\leq n}$ is bounded by $n^3 + 1$ and the equivalence of two relations can be tested in the polynomial time. Hence, $\mathscr{R}$ is a polynomial equivalence relation. The cross-composition algorithm works as follows. Given a set of malformed instances, returns some trivial no-instance of $\mathscr{P}^{\star}_{\leq 3}$-CF-*s*-*t* CUT, while given a sequence of well-formed instances, it construct a parameterized instance $(G^{\star}, H^{\star}, s, t, k+1)$ of $\mathscr{P}^{\star}_{\leq 3}$-CF-*s*-*t* CUT as follows. Let $x_i = (i-1)n+1$ and $y_i = x_i + n - 1$. Let $V(G_i) = V(H_i) = \{s_i, v_{x_i}, \cdots, v_{y_i}, t_i\}$. Now, we construct $G^{\star}$ and $H^{\star}$ as follows. $V(G^{\star}) = V(H^{\star}) = \cup_{i\in[q]}(V(G_i) \setminus \{s_i, t_i\}) \cup_{i\in[q-1]} \{w_i\} \cup \{a, s, t\}$ and $E(G^{\star}) = \cup_{i\in[q]} E(G_i - \{s_i, t_i\})$. If $s_1 v \in E(G_1)$, add sv in $E(G^{\star})$. Similarly, if

$t_q v \in E(G_q)$, add tv in $E(G^\star)$. If an edge $t_i v \in E(G_i)$ or $s_{i+1} v \in E(G_{i+1})$, add an edge $w_i v$ in $E(G^\star)$, for all $i \in [q-1]$. We also add edges sa and ta in $G^\star$. Now we define edge set of $H^\star$. $E(H^\star) = \cup_{i \in [q]} E(H_i - \{s_i, t_i\})$. We also add edge aw_i, for all $i \in [q-1]$. Since, paths are closed under vertex deletion, $H^\star$ belongs to the family $\mathscr{P}^\star_{\leq 3}$. We set parameter $k = k_1$. Figure 5.2 describes the construction of G and H. We claim that $(G^\star, H^\star, s, t, k+1)$ is a yes-instance of $\mathscr{P}^\star_{\leq 3}$-CF-$s$-$t$ CUT if and only if one of the input instance of $\mathscr{P}_{\leq 3}$-CF-s-t CUT has a conflict-free $s-t$ cut of size k.

In the forward direction, let S be a solution to $\mathscr{P}^\star_{\leq 3}$-CF-$s$-$t$ CUT in $(G^\star, H^\star, s, t, k+1)$. Since, $a \in S$, none of w_i belongs to S, where $i \in [q-1]$. We claim that $S' = (S \setminus \{a\}) \cap V(G_i)$ is a solution to $\mathscr{P}_{\leq 3}$-CF-s-t CUT in $(G_i, H_i, s_i, t_i, k_i)$, for some $i \in [q]$. Suppose not, then there exists at least one path between each pair of vertex (s_i, t_i) in G_i, where $i \in [q]$. Let P_i be a path between s_i and t_i in G_i, where $i \in [q]$. Hence, path induced by the vertex set $\cup_{i \in [q]} (V(P_i) \setminus \{s_i, t_i\}) \cup_{i \in [q-1]} \{w_i\} \cup \{s, t\}$ yields a path between s and t in $G^\star$, a contradiction. Hence, there is some G_i such that S' is a $s-t$ cut of G_i. Since $H_i[S]$ is a induced subgraph of H, $H_i[S]$ is edgeless. This completes the proof in the forward direction.

In the reverse direction, let one of the input instance $(G_i, H_i, s_i, t_i, k_i), i \in [q]$ be a yes-instance of $\mathscr{P}_{\leq 3}$-CF-s-t CUT and S be one of its solution, that is, $G_i \setminus S$ does not have a path from s_i to t_i. Clearly, by the construction of $G^\star$, $G^\star \setminus (S \cup \{a\})$ does not have a path from s to t. Since, a is not adjacent to any vertex $v \in V(H^\star) \cap V(H_i)$, where $i \in [q]$ and $uv \in E(H^\star)$ if $uv \in E(H_i)$, where $i \in [q]$, $u \neq a$ and $v \neq a$, $S \cup \{a\}$ is an independent set in H. This completes the proof. □

Theorem 5.5.3. *$\mathscr{P}^\star_{\leq 3}$-CF-$s$-$t$ CUT does not admit a polynomial compression unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Proof. Since, $\mathscr{P}_{\leq 3}$-CF-s-t CUT is NP-hard, using Lemma 32 and Theorem 5.5.2, $\mathscr{P}^\star_{\leq 3}$-CF-$s$-$t$ CUT, parameterized by the size of cut does not admit a polynomial compression

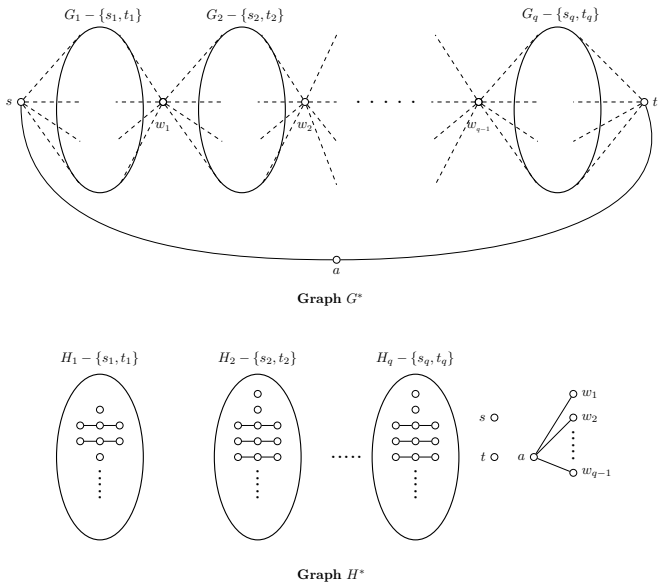

Figure 5.2: An illustration of construction of graph G and H in cross-composition from $\mathscr{P}_{\leq 3}$-CF-s-t CUT to $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT

unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$. □

Corollary 7. *$\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT does not admit a polynomial kernel.*

Proof. The proof follows from Theorem 5.5.3 and the fact that polynomial kernel is also a polynomial compression. □

5.5.3 Lower Bound for Kernel of $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT

In this subsection, we prove the main result of this section. We show that there does not exist a polynomial kernel of $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT. Towards this we give a parameter preserving reduction from $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT to $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT. Given an instance (G, H, s, t, k) of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT, we construct an instance $(G', H', k+1)$ of $\mathscr{P}^{\star\star}_{\leq 3}$-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 $x_i w_i$ and $y_i w_i$ 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

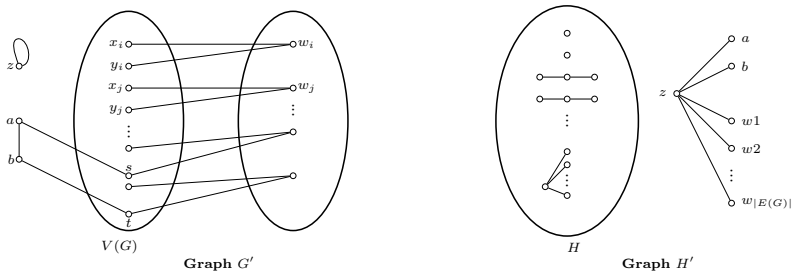

Figure 5.3: An illustration of construction of graph G and H in reduction from $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT to $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT.

$E(H') = E(H - \{s,t\})$. Additionally, we add zs, zt, za, zt, and zw_i for each $w_i \in V(H')$ to $E(H')$. Figure 5.3 describes the construction of G and H. Clearly, H' belongs to $\mathscr{P}^{\star\star}_3$ 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 $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT and $(G',H',k+1)$ of $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT in the following lemma.

Lemma 33. *(G,H,s,t,k) is a yes-instance of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT if and only if $(G',H',k+1)$ is a yes-instance of $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT.*

Proof. In the forward direction, let (G,H,s,t,k) be a yes-instance of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT and S be one of its solution. We claim that $S \cup \{z\}$ is a solution to $\mathscr{P}^{\star\star}_{\leq 3}$-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 $\mathscr{P}^{\star\star}_{\leq 3}$-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 $\mathscr{P}^{\star}_{\leq 3}$-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 5.5.4. $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT *does not admit a polynomial kernel. unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Proof. Using the construction defined above, given an instance (G, H, s, t, k) of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT, we construct an instance $(G', H', k+1)$ of $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT. Using Lemma 33, (G, H, s, t, k) is a yes-instance of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT if and only if $(G', H', k+1)$ is a yes-instance of $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT. We claim that $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT does not admit a polynomial kernel. Towards the contrary, suppose that $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT admits polynomial kernel, then the instance (G, H, s, t, k) of $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT admits a polynomial compression, a contraction to the fact $\mathscr{P}^{\star}_{\leq 3}$-CF-$s$-$t$ CUT does not admit polynomial compression unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$. □

5.6 Conclusion

In this chapter we studied kernelization complexity of $\mathscr{D}_d$-CF-FVS and $\mathscr{D}_d$-CF-OCT. We showed that the former admits a polynomial kernel of size $k^{\mathcal{O}(d)}$, while $\mathscr{D}_d$-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 $\mathscr{P}^{\star\star}_{\leq 3}$-CF-OCT. Using much more involved marking scheme we can show that $\mathscr{D}_d$-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.

Chapter 6

Parameterized Complexity of Conflict-Free Matchings and Paths

6.1 Introduction

In the previous chapters, we studied conflict-free variant of some classical combinatorial optimization problems from the viewpoint of algorithmic complexity. Recal that a typical input to a conflict-free variant of a classical problem Γ, which we call CONFLICT-FREE Γ, consists of an instance I of Γ coupled with a graph H, called the *conflict graph*. A solution to CONFLICT-FREE Γ in (I,H) is a solution to I in Γ, which is also an independent set in H. We noticed that conflict-free version of the problem introduces the constraint of "impossible pairs" in the solution that we seek for. Such a constraint of "impossible pairs" in a solution arises, for example, in the context of program testing and validation [57, 72]. Gabow et al. [57] studied the conflict-free version of finding paths in a graph, which they showed to be NP-complete.

In the previous chapters, we initiated the study of conflict-free problems in the realm of Parameterized Complexity. In particular, we studied CONFLICT-FREE $\mathscr{F}$-DELETION

problems for various families $\mathscr{F}$, of graphs such as, the family of forests, independent sets, bipartite graphs, interval graphs, etc.

MAXIMUM MATCHING and SHORTEST PATH are among the classical graph problems which are of very high theoretical and practical interest. The MAXIMUM MATCHING problem takes as input a graph G, and the objective is to compute a maximum sized subset $Y \subseteq E(G)$ such that no two edges in Y have a common vertex. MAXIMUM MATCHING is known to be solvable in polynomial time [47, 88]. The SHORTEST PATH problem takes as input a graph G and vertices s and t, and the objective is to compute a path between s and t in G with the minimum number of vertices. The SHORTEST PATH problem, together with its variants such as all-pair shortest path, single-source shortest path, weighted shortest path, etc. are known to be solvable in polynomial time [41, 9].

Darmann et al. [36] (among other problems) studied the conflict-free variants of MAXIMUM MATCHING and SHORTEST PATH. They showed that the conflict-free variant of MAXIMUM MATCHING is NP-hard even when the conflict graph is a disjoint union of edges (matching). Moreover, for the conflict-free variant of SHORTEST PATH, they showed that the problem is APX-hard, even when the conflict graph belongs to the family of 2-ladders.

In this chapter, we study conflict-free (parameterized) variants of MAXIMUM MATCHING and SHORTEST PATH, which we call CONFLICT FREE MAXIMUM MATCHING (CF-MM, for short) and CONFLICT FREE SHORTEST PATH (CF-SP, for short), respectively. We recall the formal definitions of the problems here.

CONFLICT FREE MAXIMUM MATCHING (CF-MM) **Parameter:** k
Input: A graph $G = (V,E)$, a conflict graph $H = (E,E')$, and an integer k.
Question: Is there a matching M of size at least k in G, such that M is an independent set in H?

CONFLICT FREE SHORTEST PATH (CF-SP) **Parameter:** k

Input: A graph $G = (V, E)$, a conflict graph $H = (E, E')$, two special vertices s and t, and an integer k.

Question: Is there an st-path P of length at most k in G, such that $E(P)$ is an independent set in H?

We show that both CF-MM and CF-SP are W[1]-hard, when parameterized by the solution size. The W[1]-hardness for CF-MM is obtained by giving an appropriate reduction from INDEPENDENT SET, which is known to be W[1]-hard, when parameterized by the solution size [29, 43]. In fact, our W[1]-hardness result for CF-MM holds even when the graph where we want to compute a matching is itself a matching. We show the W[1]-hardness of CF-SP by giving an appropriate reduction from a multicolored variant of the problem UNIT 2-TRACK INDEPENDENT SET (which we prove to be W[1]-hard). We note that UNIT 2-TRACK INDEPENDENT SET is known to be W[1]-hard, which is used to establish W[1]-hardness of its multicolored variant. We note that our W[1]-hardness result of CF-SP holds even when the conflict graph is a unit interval graph.

Having shown the W[1]-hardness results, we then restrict our attention to having conflict graphs belonging to some families of graphs, where the INDEPENDENT SET problem is either polynomial time solvable or solvable in FPT time. Two of the very well-known graph families that we consider are the family of chordal graphs and the family of d-degenerate graphs. For the CF-MM problem, we give an FPT algorithm, when the conflict graph belongs to the family of chordal graphs. Our algorithm is based on a dynamic programming over a "structured" tree decomposition of the conflict graph (which is chordal) together with "efficient" computation of representative families at each step of our dynamic programming routine. Notice that we cannot obtain an FPT algorithm for the CF-SP problem when the conflict graph is a chordal graph. This holds because unit-interval graphs are chordal, and the problem CF-SP is W[1]-hard, even when the conflict graph is a unit-interval graph.

For conflict graphs being d-degenerate, we obtain FPT algorithms for both CF-MM and CF-SP. These algorithms are based on the computation of an independence covering family, a notion which was recently introduced by Lokshtanov et al. [81]. We note that even for nowhere dense graphs, such an independence covering family can be computed efficiently [81]. Since our algorithms are based on computation of independence covering families, hence, our results hold even when the conflict graph is a nowhere dense graph.

Finally, we study a variant of CF-MM and CF-SP, where instead of conflicting conditions being imposed by independent sets in a conflict graph, they are imposed by independence constraints in a (representable) matroid. We give FPT algorithms for the above variant of both CF-MM and CF-SP.

6.2 Preliminaries

Sets and functions. We denote the set of natural numbers and the set of integers by $\mathbb{N}$ and $\mathbb{Z}$, respectively. By $\mathbb{N}_{\geq 1}$ we denote the set $\{x \in \mathbb{N} \mid x \geq 1\}$. For $n \in \mathbb{N}$, by $[n]$ and $[0,n]$, we denote the sets $\{1,2,\cdots,n\}$ and $\{0,1,2,\cdots,n\}$, respectively. For a set U and $p \in \mathbb{N}$, a p-family (over U) is a family of subsets of U of size p. A function $f : X \to Y$ is *injective* if for each $x,y \in X$, $f(x) = f(y)$ implies $x = y$. For a function $f : X \to Y$ and a set $S \subseteq X$, $f|_S : S \to Y$ is a function such that for $s \in S$, we have $f|_S(s) = f(s)$. We let ω denote the exponent in the running time of algorithm for matrix multiplication, the current best known bound for it is $\omega < 2.373$ [109].

Graphs. Consider a graph G. By $V(G)$ and $E(G)$ we denote the set of vertices and edges in G, respectively. For $X \subseteq V(G)$, $G[X]$ denotes the subgraph of G with vertex set X and edge set $\{uv \in E(G) \mid u,v \in X\}$. For $Y \subseteq E(G)$, $G[Y]$ denotes the subgraph of G with vertex set $\cup_{uv \in Y}\{u,v\}$ and edge set Y.

Let G be a graph. An *independent set* in G is a set $X \subseteq V(G)$ such that for every

$u, v \in X$, $uv \notin E(G)$. A *matching* in G is a set $Y \subseteq E(G)$ such that no two distinct edges in Y have a common vertex. A matching M in G is a *maximum matching* if for any matching Y in G, $|M| \geq |Y|$. A matching M in G saturates a set $X \subseteq V(G)$, if every vertex in X is an end point of an edge in M. For $v_1, v_\ell \in V(G)$, a *v_1v_ℓ-path* $P = (v_1, v_2, \cdots, v_{\ell-1}, v_\ell)$ in G is a sequence of (distinct) vertices, such that $V(P) \subseteq V(G)$ and for each $i \in [\ell - 1]$, we have $v_iv_{i+1} \in E(G)$. Moreover, the edges in $\{v_iv_{i+1} \mid i \in [\ell - 1]\}$ are called edges in P. The *length* of a path is the number of edges in it. A *shortest uv-path* is a uv-path with minimum number of edges.

A *chordal graph* is a graph with no induced cycles of length at least four. An *interval graph* is an intersection graph of line segments (intervals) on the real line, that is, its vertex set is a set of intervals, and two vertices are adjacent if and only if their corresponding intervals intersect. A *unit-interval* graph is an intersection graph of intervals of unit length on the real line. For $d \in \mathbb{N}$, a graph is *d-degenerate* if every subgraph of it has a vertex of degree at most d. A *clique* K in G is an (induced) subgraph, such that for any two distinct vertices $u, v \in V(K)$ we have $uv \in E(G)$. A vertex set $S \subseteq V(G)$ is a *clique* in G if $G[S]$ is a clique. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. If $V_1 \cap V_2 = \emptyset$, then disjoint union of G_1 and G_2 is the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$. If $V_1 = V_2$, then the edge-wise union of G_1 and G_2 is the graph $G = (V_1, E_1 \cup E_2)$.

In the following we state definitions related to tree decomposition and some results on them, that are used in our algorithms.

Definition 18. A *tree decomposition* of a graph H is a pair (T, X), where T is a rooted tree and $X = \{X_t \mid t \in V(T)\}$. Every node t of T is assigned a subset $X_t \subseteq V(H)$, called a *bag*, such that following conditions are satisfied:

- $\bigcup_{t \in V(T)} X_t = V(H)$, that is, each vertex in H is in at least one bag;
- For every edge $uv \in E(H)$, there is $t \in V(T)$ such that $u, v \in X_t$;
- For every vertex $v \in V(H)$ the graph $T[\{t \in V(T) \mid v \in X_t\}]$ is a connected subtree

of T.

To distinguish between vertices of a graph H and vertices of its tree decomposition (T,X), we refer to the vertices in T as *nodes*. Since T is a rooted tree, we have a natural parent-child and ancestor-descendant relationship among nodes in T. For a node $t \in V(T)$, by $\mathsf{desc}(t)$ we denote the set descendant of t in T (including t). For a node $t \in V(T)$ by V_t we denote the union of all bags in the subtree rooted at t, that is, $V_t = \cup_{d \in \mathsf{desc}(t)} X_d$ and by H_t we denote the graph $H[V_t]$. A *leaf* node of T is a node with degree exactly one in T, which is different from the root node. All the nodes of T which are neither the root node nor a leaf node are *non-leaf* nodes.

We now define a more structured form of tree decomposition that will be used in the algorithm.

Definition 19. Let (T,X) be a tree decomposition of a graph H with r as the root node. Then, (T,X) is a *nice tree decomposition* if for each each leaf ℓ in T and the root r, we have that $X_\ell = X_r = \emptyset$, and each non-leaf node $t \in V(T)$ is of one of the following types:

1. **Introduce node:** t has exactly one child, say t', and $X_t = X_{t'} \cup \{v\}$, where $v \notin X_{t'}$. We say that v is *introduced* at t;
2. **Forget node:** t has exactly one child, say t', and $X_t = X_{t'} \setminus \{v\}$, where $v \in X_{t'}$. We say that v is *forgotten* at t;
3. **Join node:** t has exactly two children, say t_1 and t_2, and $X_t = X_{t_1} = X_{t_2}$.

Proposition 14. [[29, 69]] *Given a tree decomposition (T,X) of a graph H, in polynomial time we can compute a nice tree decomposition (T',X') of H that has at most $\mathcal{O}(k|V(H)|)$ nodes, where, k is the size of the largest bag in X. Moreover, for each $t' \in V(T')$, there is $t \in V(T)$ such that $X'_{t'} \subseteq X_t$.*

A tree decomposition (T,X) of a graph H, where for each $t \in V(T)$, the graph $H[X_t]$

is a clique, is called a *clique-tree*. Next, we state a result regarding computation of a clique-tree of a chordal graph.

Proposition 15. [[58]] *Given an n vertex chordal graph H, in polynomial time we can construct a clique-tree (T,X) of H with $\mathcal{O}(n)$ nodes.*

Using Proposition 14 and 15 we obtain the following result.

Proposition 16. *Given an n vertex chordal graph H, in polynomial time we can construct a nice tree decomposition which is also a clique-tree (nice clique-tree), (T,X) of H with $\mathcal{O}(n^2)$ nodes.*

Matroids and representative sets. In the following we state some basic definitions related to matroids. We refer the reader to [97] for more details. We also state the definition of representative families and state some results related to them.

Definition 20. A pair $\mathscr{M} = (U, \mathscr{I})$, where U is the ground set and $\mathscr{I}$ is a family of subsets of U, is a *matroid* if the following conditions hold:

- $\emptyset \in \mathscr{I}$;
- If $I_1 \in \mathscr{I}$ and $I_2 \subseteq I_1$, then $I_2 \in \mathscr{I}$;
- If $I_1, I_2 \in \mathscr{I}$ and $|I_2| < |I_1|$, then there exists an element $x \in I_1 \setminus I_2$, such that $I_2 \cup \{x\} \in \mathscr{I}$.

An inclusion-wise maximal set in $\mathscr{I}$ is called a *basis* of $\mathscr{M}$. All bases of a matroid are of the same size. The size of a basis is called the *rank* of the matroid. For a matroid $\mathscr{M} = (U, \mathscr{I})$ and sets $P, Q \subseteq U$, we say that P *fits* Q if $P \cap Q = \emptyset$ and $P \cup Q \in \mathscr{I}$.

A matroid $\mathscr{M} = (U, \mathscr{I})$ is a *linear* (or *representable*) matroid if there is a matrix A over a field $\mathbb{F}$ with E as the set of columns, such that: 1) $|E| = |U|$; 2) there is an injective function $\varphi : U \to E$, such that $X \subseteq U$ is an independent set in $\mathscr{M}$ if and only if

$\{\varphi(x) \mid x \in X\}$ is a set of linearly independent columns (over $\mathbb{F}$). In the above, we say that $\mathscr{M}$ is representable over $\mathbb{F}$, and A is one of its representation.

In the following, we define some matroids and state results regarding computation of their representations.

Definition 21 ([29, 97])**.** A matroid $\mathscr{M} = (U, \mathscr{I})$ is a *partition* matroid if the ground set U is partitioned into sets $U_1, U_2, \cdots, U_k$, and for each $i \in [k]$, there is an integer a_i associated with U_i. A set $S \subseteq U$ is independent in $\mathscr{M}$ if and only if for each $i \in [k]$, $|S \cap U_i| \leq a_i$.

Proposition 17. [[52, 97, 85]] *A representation of a partition matroid over $\mathbb{Q}$ (the field of rationals) can be computed in polynomial time.*

Definition 22. Let $\mathscr{M}_1 = (U_1, \mathscr{I}_1), \mathscr{M}_2 = (U_2, \mathscr{I}_2) \cdots, \mathscr{M}_t = (U_t, \mathscr{I}_t)$ be t matroids with $U_i \cap U_j = \emptyset$, for all $1 \leq i \neq j \leq t$. The *direct sum* $\mathscr{M}_1 \oplus \cdots \oplus \mathscr{M}_t$, of $\mathscr{M}_1, \mathscr{M}_2, \cdots, \mathscr{M}_t$ is the matroid with ground set $U = \cup_{i \in [t]} U_i$ and $X \subseteq U$ is independent in $\mathscr{M}$ if and only if for each $i \in [t]$, $X \cap U_i \in \mathscr{I}_i$.

Proposition 18. [[85, 97]] Given matrices $A_1, A_2, \cdots, A_t$ (over $\mathbb{F}$) representing matroids $\mathscr{M}_1, \mathscr{M}_2, \cdots, \mathscr{M}_t$, respectively, we can compute a representation of their direct sum, $\mathscr{M}_1 \oplus \cdots \oplus \mathscr{M}_t$, in polynomial time.

Next, we state the definition of representative families.

Definition 23. Let $\mathscr{M} = (U, \mathscr{I})$ be a matroid, and $\mathscr{A}$ be a p-family of U. We say that $\mathscr{A}' \subseteq \mathscr{A}$ is a q-representative for $\mathscr{A}$ if for every set $Y \subseteq U$ of size q, if there is a set $X \in \mathscr{A}$, such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathscr{I}$, then there is a set $X' \in \mathscr{A}'$ such that $X' \cap Y = \emptyset$ and $X' \cup Y \in \mathscr{I}$. If $\mathscr{A}' \subseteq \mathscr{A}$ is a q-representative for $\mathscr{A}$ then we denote it by $\mathscr{A}' \subseteq^q_{rep} \mathscr{A}$.

In the following, we state some basic propositions regarding q-representative sets, which will be used later.

Proposition 19. [[29, 52]] *If $\mathscr{A}_1 \subseteq^q_{\mathsf{rep}} \mathscr{A}_2$ and $\mathscr{A}_2 \subseteq^q_{\mathsf{rep}} \mathscr{A}_3$, then $\mathscr{A}_1 \subseteq^q_{\mathsf{rep}} \mathscr{A}_3$.*

Proposition 20. [[29, 52]] *If $\mathscr{A}_1$ and $\mathscr{A}_2$ are two p-families such that $\mathscr{A}_1' \subseteq^q_{\mathsf{rep}} \mathscr{A}_1$ and $\mathscr{A}_2' \subseteq^q_{\mathsf{rep}} \mathscr{A}_2$, then $\mathscr{A}_1' \cup \mathscr{A}_2' \subseteq^q_{\mathsf{rep}} \mathscr{A}_1 \cup \mathscr{A}_2$.*

Next, we state a result regarding the computation of a q-representative set.

Theorem 6.2.1 ([29, 52])**.** *Given a matrix M (over field $\mathbb{F}$) representing a matroid $\mathscr{M} = (U, \mathscr{I})$ of rank k, a p-family $\mathscr{A}$ of independent sets in $\mathscr{M}$, and an integer q such that $p+q=k$, there is an algorithm which computes a q-representative family $\mathscr{A}' \subseteq^q_{\mathsf{rep}} \mathscr{A}$ of size at most $\binom{p+q}{p}$ using at most $\mathscr{O}\big(|\mathscr{A}|\big(\binom{p+q}{p}p^{\omega} + \binom{p+q}{p}^{\omega-1}\big)\big)$ operations over $\mathbb{F}$.*

Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be two families of sets over U and $\mathscr{M} = (U, \mathscr{I})$ be a matroid. We define their convolution as follows.

$$\mathscr{A}_1 \star \mathscr{A}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathscr{A}_1, A_2 \in \mathscr{A}_2, A_1 \cap A_2 = \emptyset \text{ and } A_1 \cup A_2 \in \mathscr{I}\}$$

Lemma 34. *Let $\mathscr{M} = (U, \mathscr{I})$ be a matroid, $\mathscr{A}_1$ be a p_1-family, and $\mathscr{A}_2$ be a p_2-family. If $\mathscr{A}_1' \subseteq^{k-p_1}_{\mathsf{rep}} \mathscr{A}_1$ and $\mathscr{A}_2' \subseteq^{k-p_2}_{\mathsf{rep}} \mathscr{A}_2$, then $\mathscr{A}_1' \star \mathscr{A}_2' \subseteq^{k-p_1-p_2}_{\mathsf{rep}} \mathscr{A}_1 \star \mathscr{A}_2$.*

Proof. The proof of this lemma is similar to the proof of Lemma 12.28 in [29]. Let B be a set of size $k-p_1-p_2$. Suppose there exists a set $A_1 \cup A_2 \in \mathscr{A}_1 \star \mathscr{A}_2$ that fits B. Since, $(A_1 \cup A_2) \cap B = \emptyset$, we have $|B \cup A_2| = k - p_1$. This implies that there exists $A_1' \in \mathscr{A}_1'$ which fits $B \cup A_2$, that is, $(A_1' \cup B \cup A_2) \in \mathscr{I}$ and $A_1' \cap (B \cup A_2) = \emptyset$ which gives $|A_1' \cup B| = k - p_2$. This means, there exists $A_2' \in \mathscr{A}_2'$ that fits $A_1' \cup B$, that is, $(A_2' \cup A_1' \cup B) \in \mathscr{I}$ and $A_2' \cap (A_1' \cup B) = \emptyset$. Since, $A_1' \cap (B \cup A_2) = \emptyset$ and $A_2' \cap (A_1' \cup B) = \emptyset$, we get $(A_1' \cup A_2') \cap B = \emptyset$. Hence, $A_1' \cup A_2'$ fits B and $(A_1' \cup A_2') \in \mathscr{A}_1' \star \mathscr{A}_2'$. □

Next, we give a result regarding computation of convolution ($\star$).

Proposition 21. *Let M be a matrix over a field $\mathbb{F}$ representing a matroid $\mathscr{M} = (U, \mathscr{I})$ over an n-element ground set, $\mathscr{A}_1$ be a p_1-family, and $\mathscr{A}_2$ be a p_2-family, where $p_1 + p_2 = k$. Then $\mathscr{A}_1 \star \mathscr{A}_2$ can be computed in time $\mathscr{O}(2^k n^{\mathscr{O}(1)})$.*

Proof. Consider the standard convolution operation, $\mathscr{A}_1 \circ \mathscr{A}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathscr{A}_1, A_2 \in \mathscr{A}_2 \text{ and } A_1 \cap A_2 = \emptyset\}$ defined in [29, Section 12.3.5]. The family $\mathscr{A}_1 \circ \mathscr{A}_2$ can be computed in $\mathcal{O}(2^k n^3)$ time [29, Exercise 12.12]. Since, $\mathscr{A}_1 \star \mathscr{A}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathscr{A}_1, A_2 \in \mathscr{A}_2, A_1 \cap A_2 = \emptyset, \text{ and } A_1 \cup A_2 \in \mathscr{I}\}$. Hence, $X \in \mathscr{A}_1 \star \mathscr{A}_2$ if and only if $X \in \mathscr{A}_1 \circ \mathscr{A}_2$ and X is a set of linearly independent columns (over $\mathbb{F}$). Testing whether a set of vectors is linearly independent over a field can be done in time polynomial in size of the set (using Gaussian elimination). Therefore, testing if an $X \in \mathscr{A}_1 \circ \mathscr{A}_2$ is linearly independent, can be done in time $\mathcal{O}(n^{\mathcal{O}(1)})$. Since $|\mathscr{A}_1 \circ \mathscr{A}_2| \leq |\mathscr{A}_1||\mathscr{A}_2|$, family $\mathscr{A}_1 \star \mathscr{A}_2$ can be computed in $\mathcal{O}((2^k + |\mathscr{A}_1||\mathscr{A}_2|)n^{\mathcal{O}(1)})$ time. Since, $|\mathscr{A}_1| \leq 2^{p_1}$ and $|\mathscr{A}_2| \leq 2^{p_2}$, the running time is bounded by $\mathcal{O}(2^k n^{\mathcal{O}(1)})$. □

Universal sets and their computation.

Definition 24. An (n,k)-*universal set* is a family $\mathscr{F}$ of subsets of $[n]$ such that for any set $S \subseteq [n]$ of size k, the family $\{A \cap S \mid A \in \mathscr{F}\}$ contains all 2^k subsets of S.

Next, we state a result regarding the computation of a universal set.

Proposition 22. [29, 93] *For any $n,k \geq 1$, we can compute an (n,k)-universal set of size $2^k k^{\mathcal{O}(\log k)} \log n$ in time $2^k k^{\mathcal{O}(\log k)} n \log n$.*

6.3 W[1]-hardness Results

In this section, we show that CONFLICT FREE MAXIMUM MATCHING and CONFLICT FREE SHORTEST PATH are W[1]-hard, when parameterized by the solution size.

6.3.1 W[1]-hardness of CF-MM

We show that CF-MM is W[1]-hard, when parameterized by the solution size, even when the graph where we want to find a matching, is itself a matching (disjoint union of edges). To prove our result, we give an appropriate reduction from INDEPENDENT SET to CF-MM. In the following, we define the problem INDEPENDENT SET.

INDEPENDENT SET	**Parameter:** k
Input: A graph G and an integer k.	
Question: Is there a set $X \subseteq V(G)$ of size at least k such that X is an independent set in G?	

It is known that INDEPENDENT SET is W[1]-hard, when parameterized by the size of an independent set [29, 43].

Theorem 6.3.1. CF-MM *is* W[1]*-hard, when parameterized by the solution size.*

Proof. Given an instance $(G^\star, k)$ of INDEPENDENT SET, we construct an equivalent instance (G, H, k) of CF-MM as follows. We first describe the construction of G. For each $v \in V(G^\star)$, we add an edge vv' to G. Notice that G is a matching. This completes the description of G. Next, we move to the construction of H. We have $V(H) = \{e_v = vv' \mid v \in V(G^\star)\}$. Moreover, for $e_u, e_v \in V(H)$, we add the edge $e_u e_v$ to $E(H)$ if and only if $uv \in E(G^\star)$. We note that H is exactly the same as $G^\star$, with vertices being renamed. This completes the construction of (G, H, k) of CF-MM. Next, we show that $(G^\star, k)$ is a yes-instance of INDEPENDENT SET if and only if (G, H, k) is a yes-instance of CF-MM.

In forward direction, let $(G^\star, k)$ be a yes-instance of INDEPENDENT SET, and S be one of its solution. Let $S' = \{e_v \mid v \in S\}$. We show that S' is a solution to CF-MM. Notice that by construction, S' is a matching in G, and $|S'| = |S| \geq k$. Moreover, $G^\star$ is isomorphic to H, with the vertex mapping as $\varphi : V(G^\star) \to V(H)$, where for $v \in V(G^\star)$, $\varphi(v) = e_v$. Hence, S' is an independent set in H.

In reverse direction, let (G,H,k) be a yes-instance of CF-MM, and S' be one of its solution. Let $S = \{v \mid e_v \in S'\}$. Using an analogous argument as in the forward direction, we conclude that S is a solution to INDEPENDENT SET in $(G^\star,k)$. This concludes the proof. □

6.3.2 W[1]-hardness of CF-SP

We show that CF-SP is W[1]-hard, when parameterized by the solution size, even when the conflict graph is a proper interval graph. We refer to this restricted variant of the problem as UNIT INTERVAL CF-SP. To prove our result, we give an appropriate reduction from a multicolored variant of the problem UNIT 2-TRACK INDEPENDENT SET, which we call UNIT 2-TRACK MULTICOLORED IS. In the following, we define the problems UNIT 2-TRACK INDEPENDENT SET and UNIT 2-TRACK MULTICOLORED IS.

UNIT 2-TRACK INDEPENDENT SET (UNIT 2-TRACK IS) **Parameter:** k

Input: Two unit-interval graphs $G_1 = (V,E_1)$ and $G_2 = (V,E_2)$, and an integer k.

Question: Is there a set $S \subseteq V$ of size at least k, such that S is an independent set in both G_1 and G_2?

UNIT 2-TRACK MULTICOLORED IS (UNIT 2-TRACK MIS)

Input: Two unit-interval graphs $G_1 = (V,E_1)$ and $G_2 = (V,E_2)$, and a partition $V_1, V_2, \cdots, V_k$ of V.

Parameter: k

Question: Is there a set $S \subseteq V$, such that S is an independent set in both G_1 and G_2, and for each $i \in [k]$, we have $|S \cap V_i| = 1$?

It is known that UNIT 2-TRACK IS is W[1]-hard, when parameterized by the solution size [65]. We show that the problem UNIT 2-TRACK MIS is W[1]-hard, when parameterized by the number of sets in the partition. We show this by giving an appropriate (Turing) reduction from UNIT 2-TRACK IS. Finally, we give a reduction from UNIT 2-TRACK

MIS to UNIT INTERVAL CF-SP, hence obtaining the desired result.

6.3.3 W[1]-hardness of UNIT 2-TRACK MIS.

We give a (Turing) reduction from UNIT 2-TRACK IS to UNIT 2-TRACK MIS. Moreover, since we want to rule out existence of an FPT algorithm, we spend FPT time to obtain FPT many instances of UNIT 2-TRACK MIS.

Before proceeding to the reduction from UNIT 2-TRACK IS to UNIT 2-TRACK MIS, we define the notion of *perfect hash family*, which will be used in the reduction.

Definition 25. An (n,k)-*perfect hash family* $\mathscr{F}$, is a family of functions $f : [n] \to [k]$ such that for every set $S \subseteq [n]$ of size k, there is an $f \in \mathscr{F}$, such that $f|_S$ is injective.

In the following, we state a result regarding computation of an (n,k)-perfect hash family.

Theorem 6.3.2. [29, 93] *For any $n,k \geq 1$, an (n,k)-perfect hash family of size $e^k k^{\mathscr{O}(\log k)} \log n$ can be constructed in $e^k k^{\mathscr{O}(\log k)} n \log n$ time.*

Now we are ready to give a (Turing) reduction from UNIT 2-TRACK IS to UNIT 2-TRACK MIS.

Lemma 35. *There is a parameterized Turing reduction from* UNIT 2-TRACK IS *to* UNIT 2-TRACK MIS.

Proof. Let (G_1, G_2, k) be an instance of UNIT 2-TRACK IS, where $V(G_1) = V(G_2) = [n]$. We construct a family $\mathscr{C}$ of instances of UNIT 2-TRACK MIS as follows. We start by computing an (n,k)-perfect hash family $\mathscr{F}$, of size $e^k k^{\mathscr{O}(\log k)} \log n$, in time $e^k k^{\mathscr{O}(\log k)} n \log n$, using Theorem 6.3.2. Now, for each $f \in \mathscr{F}$, we construct an instance $I_f = (G_1, G_2, V_1^f, V_2^f, \cdots, V_k^f)$ of UNIT 2-TRACK MIS as follows. For $i \in [k]$, we set $V_i^f = \{v \in V(G_1) \mid f(v) = i\}$. Finally, we set $\mathscr{C} = \{I_f \mid f \in \mathscr{F}\}$.

We claim that (G_1, G_2, k) is a yes-instance of UNIT 2-TRACK IS if and only if there is $I_f \in \mathscr{C}$ such that I_f is a yes-instance of UNIT 2-TRACK MIS.

In the forward direction, let (G_1, G_2, k) be a yes-instance of UNIT 2-TRACK IS, and S be one of its solution of size k. Consider $f \in \mathscr{F}$ such that $f|_S$ is injective, which exists since $\mathscr{F}$ is an (n,k)-perfect hash family. By construction of $\mathscr{C}$, we have $I_f \in \mathscr{C}$. Moreover, by construction of f, for each $i \in [k]$, we have $|S \cap V_i| = 1$. Hence, S is a solution to I_f.

In the reverse direction, let $I_f \in \mathscr{C}$ be a yes-instance of UNIT 2-TRACK MIS, and S be one of its solution. Clearly, S is a solution to UNIT 2-TRACK IS in (G_1, G_2, k) as $I_f = (G_1, G_2, V_1^f, V_2^f, \cdots, V_k^f)$. This concludes the proof. □

Theorem 6.3.3. UNIT 2-TRACK MIS *is* W[1]-hard*, when parameterized by the solution size.*

Proof. Follows from Lemma 35 and W[1]-hardness of UNIT 2-TRACK IS. □

6.3.4 W[1]-hardness of UNIT INTERVAL CF-SP

We give a parameterized reduction from UNIT 2-TRACK MIS to UNIT INTERVAL CF-SP. Let $(G_1, G_2, V_1, \cdots, V_k)$ be an instance of UNIT 2-TRACK MIS. We construct an instance (G', H, s, t, k') of UNIT INTERVAL CF-SP as follows. For each $v \in V(G_1)$, we add a path on 3 vertices namely, (v_1, v_2, v_3) in G'. For notational convenience, for $v \in V(G_1)$, by $e_{12}(v)$ and $e_{23}(v)$ we denote the edges v_1v_2 and v_2v_3, respectively. Consider $i \in [k-1]$. For $u \in V_i$ and $v \in V_{i+1}$, we add the edge $z_{uv} = u_3v_1$ to $E(G')$ (see Figure 6.1). Moreover, by Z_i, we denote the set $\{z_{uv} \mid u \in V_i, v \in V_{i+1}\}$. We add two new vertices s and t to $V(G')$, and add all the edges in $Z_0 = \{sv_1 \mid v \in V_1\}$ and $Z_k = \{v_3t \mid v \in V_k\}$ to $E(G')$. Next, we move to the construction of H. Note that H must be a unit-interval graph on the vertex set $E(G') = (\cup_{i \in [0,k]} Z_i) \cup (\cup_{v \in V(G_1)} \{e_{12}(v), e_{23}(v)\})$. In H, each vertex in $\cup_{i \in [0,k]} Z_i$ is an isolated vertex. Let $E_{12} = \{e_{12}(v) \mid v \in V(G_1)\}$ and $E_{23} = \{e_{23}(v) \mid v \in V(G_1)\}$. For $e_{12}(u), e_{12}(v) \in E_{12}$, we add the edge $e_{12}(u)e_{12}(v)$ to $E(H)$ if and only if $uv \in E(G_1)$.

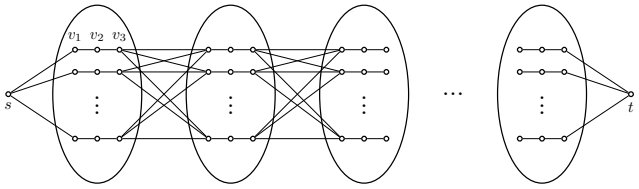

Figure 6.1: An illustration of the construction of G' in W[1]-hardness of UNIT INTERVAL CF-SP.

Similarly, for $e_{23}(u), e_{23}(v) \in E_{23}$, we add the edge $e_{23}(u)e_{23}(v)$ to $E(H)$ if and only if $uv \in E(G_2)$. Observe that $H[E_{12}]$ is isomorphic to G_1, with bijection $\phi_1 : V(G_1) \to E_{12}$ with $\phi_1(v) = e_{12}(v)$. Similarly, $H[E_{23}]$ is isomorphic to G_2 with bijection $\phi_2 : V(G_2) \to E_{23}$ with $\phi_2(v) = e_{23}(v)$. By construction, H is a disjoint union of unit-interval graphs, and hence is a unit-interval graph. Finally, we set $k' = 3k+1$. This completes the description of the reduction.

In the following lemma we show that the instance $(G_1, G_2, V_1, \cdots, V_k)$ of UNIT 2-TRACK MIS and the instance (G', H, s, t, k') of UNIT INTERVAL CF-SP are equivalent.

Lemma 36. $(G_1, G_2, V_1, \cdots, V_k)$ *is a* yes*-instance of* UNIT 2-TRACK MIS *if and only if* (G', H, s, t, k') *is a* yes*-instance of* UNIT INTERVAL CF-SP.

Proof. In the forward direction, let $(G_1, G_2, V_1, \cdots, V_k)$ be a yes-instance of UNIT 2-TRACK MIS, and $S = \{v^1, v^2, \cdots, v^k\}$ be one of its solution, such that $v^i \in V_i$. We claim that $P = (s, v_1^1, v_2^1, v_3^1, \cdots, v_1^k, v_2^k, v_3^k, t)$ is a conflict-free path (on $3k+1$ edges) in G'. By the construction of G', it follows that P is a path in G'. Next, we show that $E(P)$ is an independent set in H. Let $v_3^0 = s$ and $v_1^{k+1} = t$. By construction, each edge in $\{v_3^i v_1^{i+1} \mid i \in [0,k]\} \subseteq \cup_{[0,k]} Z_i$ is an isolated vertex in H. Also, for each $i \in [k]$, we have that $\{e_{12}(v^i), e_{23}(v^i)\}$ is an independent set in H. Next, consider $i, j \in [k]$, where $i \neq j$. By construction $e_{12}(v^i)e_{23}(v^j), e_{23}(v^i)e_{12}(v^j) \notin E(H)$. Moreover, $e_{12}(v^i)e_{12}(v^j) \notin E(H)$ since S in an independent set in G_1. Similarly, $e_{23}(v^i)e_{23}(v^j) \notin E(H)$ as S is an independent set in G_2. In the above, we have considered every pair of edges in $E(P)$, and argued that no two of them are adjacent to each other in H. Hence, it follows that P is a solution to UNIT

INTERVAL CF-SP in (G', H, s, t, k').

In the reverse direction, let P be a solution to UNIT INTERVAL CF-SP in (G', H, s, t, k'). By the construction of G', the path P must be of the form $(s, v_1^1, v_2^1, v_3^1, \cdots, v_1^k, v_2^k, v_3^k, t)$. We claim that $S = \{v^1, v^2, \cdots, v^k\}$ is an independent set in both G_1 and G_2. Suppose not, then there is an edge $v^i v^j$, $i \neq j$ and $i, j \in [k]$ say, in G_1 (the case when it is in G_2 is symmetric). But then $e_{12}(v^i)e_{12}(v^j)$ is an edge in H, contradicting that $E(P)$ is an independent set in H. Hence, we have that S is an independent set both in G_1 and G_2. Moreover, since P is a path of length at most $3k+1$, it must hold that for each $i \in [k]$, we have $v^i \in V_i$. Hence, S is a solution to UNIT 2-TRACK MIS in $(G_1, G_2, V_1, \cdots, V_k)$. □

Theorem 6.3.4. UNIT INTERVAL CF-SP *is* W[1]-hard*, when parameterized by the solution size.*

Proof. Follows from the construction of instance (G', H, s, t, k') of UNIT INTERVAL CF-SP, for the given instance $(G_1, G_2, V_1, \cdots, V_k)$ of UNIT 2-TRACK MIS, Lemma 36, and Theorem 6.3.3. □

6.4 FPT Algorithm for CF-MM with Chordal Conflict

In this section, we show that CF-MM is FPT, when the conflict graph belongs to the family of chordal graphs. We call this restricted version of CF-MM as CHORDAL CONFLICT MATCHING. Towards designing an algorithm for CHORDAL CONFLICT MATCHING, we first give an FPT algorithm for a restricted version of CHORDAL CONFLICT MATCHING, where we want to compute a matching for a bipartite graph. We call this variant of CHORDAL CONFLICT MATCHING as CHORDAL CONFLICT BIPARTITE MATCHING (CCBM). We then employ the algorithm for CCBM to design an FPT algorithm for CHORDAL CONFLICT MATCHING.

6.4.1 FPT algorithm for CCBM

We design an FPT algorithm for the problem CCBM, where the conflict graph is chordal and the graph where we want to compute a matching is a bipartite graph. The problem CCBM is formally defined below.

CHORDAL CONFLICT BIPARTITE MATCHING (CCBM) **Parameter:** k
Input: A bipartite graph $G = (V,E)$ with vertex bipartition L,R, a conflict graph $H = (E,E')$, and an integer k.
Question: Is there a matching $M \subseteq E$ of size k in G, such that M is an independent set in H?

The FPT algorithm for CCBM is based on a dynamic programming routine over a tree decomposition of the conflict graph H and representative sets on the graph G. Let (G,L,R,H,k) be an instance of CF-MM, where G is a bipartite graph on n vertices, with vertex bipartition L,R, and H is a chordal graph with $V(H) = E(G)$.

In the following, we construct three matroids $\mathscr{M}_L = (E, \mathscr{I}_L), \mathscr{M}_R = (E^c, \mathscr{I}_R)$, and $\mathscr{M} = (E \cup E^c, \mathscr{I})$. Matroids $\mathscr{M}_L$ and $\mathscr{M}_R$ are partition matroids and the matroid $\mathscr{M}$ is the direct sum of $\mathscr{M}_L$ and $\mathscr{M}_R$. The ground set of $\mathscr{M}_L$ is $E = E(G)$. The set E^c contains a copy of edges in E, that is, $E^c = \{e^c \mid e \in E\}$. We create two (disjoint) sets E and E^c, because $\mathscr{M}$ is the direct sum of $\mathscr{M}_L$ and $\mathscr{M}_R$, and we want their ground sets to be disjoint. Next, we describe the partition $\mathscr{E}$ of E into $|L|$ sets and $|L|$ integers, one for each set in the partition, for the partition matroid $\mathscr{M}_L$. For $u \in L$, let $E_u = \{uv \mid uv \in E\}$. Notice that for $u,v \in L$, where $u \neq v$, we have $E_u \cap E_v = \emptyset$. Moreover, $\cup_{u \in E} E_u = E$. We let $\mathscr{E} = \{E_u \mid u \in L\}$, and for each $u \in L$, we set $a_u = 1$. Similarly, we define the partition $\mathscr{E}^c$ of E^c with respect to set R. That is, we let $\mathscr{E}^c = \{E^c_u = \{(uv)^c \mid uv \in E(G)\} \mid u \in R\}$. Furthermore, for $u \in R$, we let $a_{u^c} = 1$. We define the following notation, which will be used later. For $Z \subseteq E$, we let $Z^c = \{e^c \mid e \in Z\} \subseteq E^c$.

Proposition 23. *$Q \subseteq E(G)$ is a matching in G with vertex bipartition L and R if and only*

if $Q \cup Q^c$ is an independent set in the matroid $\mathscr{M} = \mathscr{M}_L \oplus \mathscr{M}_R$.

Proof. In the forward direction, let Q be a matching in the bipartite graph $G = (V, E)$, where $V = L \cup R$. Since, $\mathscr{M}_L = (E, \mathscr{I}_L)$ is a partition matroid with partition $\mathscr{E} = \{E_u \mid u \in L\}$ and $a_u = 1$, for each $u \in L$, $Q \cap L$ is an independent set in $\mathscr{M}_L$. Similarly, $Q^c \cap R$ is an independent in $\mathscr{M}_R$. Since, $\mathscr{M} = \mathscr{M}_L \oplus \mathscr{M}_R$, it follows that $Q \cup Q^c$ is an independent set in $\mathscr{M}$.

In the reverse direction, consider $Q \subseteq E$ such that $Q \cup Q^c$ is an independent set in $\mathscr{M}$. Since, $\mathscr{M} = \mathscr{M}_{\mathscr{L}} \oplus \mathscr{M}_{\mathscr{R}}$, Q is independent in $\mathscr{M}_L$ and Q^c is independent in $\mathscr{M}_R$. Since, Q and Q^c both have copies of the same edge, no two edges in Q share an end point in G. Hence, Q forms a matching in G. □

To capture the independence property on the conflict graph, we rely on the fact that a chordal graph admits a nice clique-tree (Proposition 16). This allows us to do dynamic programming over a nice clique-tree. At each step of our dynamic programming routine, using representative sets, we ensure that we store a family of sets which are enough to recover (some) independent set in $\mathscr{M}$, if a solution exists.

We now move to the formal description of the algorithm. The algorithm starts by computing a nice clique-tree (T, X) of H in polynomial time, using Proposition 16. Let $r \in V(T)$ be the root of the (rooted) tree T. For $X_t \in X$, we let $\mathscr{X}_t = \{\emptyset\} \cup \{\{v\} \mid v \in X_t\}$. Recall that for $t \in V(T)$, H_t is the graph $H[V_t]$, where $V_t = \cup_{d \in \mathsf{desc}(t)} X_d$.

In the following, we state some notations, which will be used in the algorithm. For each $t \in V(T)$, $Y \in \mathscr{X}_t$, and an integer $p \in [0, k]$ we define a family $\mathscr{P}^p_{t,Y}$ as follows.

$$\mathscr{P}^p_{t,Y} = \{Z \cup Z^c \mid Z \subseteq V(H_t)(\subseteq E), |Z| = p, Z \cap X_t = Y, Z \cup Z_c \in \mathscr{I} \text{ and } H_t[Z] \text{ is edgeless}\}$$

For a family $\mathscr{F}$ of subsets of $E \cup E^c$, $\mathscr{F}$ is called a *paired-family* if for each $F \in \mathscr{F}$, there is $Z \subseteq E$, such that $F = Z \cup Z^c$.

In the following definition, we state the entries in our dynamic programming routine.

Definition 26. *For each $t \in V(T)$, $Y \in \mathscr{X}_t$ and $p \in [0,k]$, we have an entry $c[t,Y,p]$, which stores a paired-family $\mathscr{F}(t,Y,p)$ of subsets of $E \cup E^c$ of size $2p$, such that for each $F = Z \cup Z^c \in \mathscr{F}$, the following conditions are satisfied.*

1. *$|Z| = p$;*
2. *$Z \cap X_t = Y$;*
3. *Z is a matching in G, that is, Z and Z^c are independent sets in $\mathscr{M}_L$ and $\mathscr{M}_R$, respectively;*
4. *Z is an independent set in H_t.*

Moreover, $\mathscr{F} \neq \emptyset$ if and only if $\mathscr{P}^p_{t,Y} \neq \emptyset$.

Consider $t \in V(T)$, $Y \in \mathscr{X}_t$ and $p \in [0,k]$. Observe that $\mathscr{P}^p_{t,Y}$ is a valid candidate for $c[t,Y,p]$, which also implies that (G,H,k) is a yes-instance of CCBM if and only if $c[r,\emptyset,k] \neq \emptyset$. However, we cannot set $c[t,Y,p] = \mathscr{P}^p_{t,Y}$ as the size of $\mathscr{P}^p_{t,Y}$ could be exponential in n, and the goal here is to obtain an FPT algorithm. Hence, we will store a much smaller subfamily (of size at most $\binom{2k}{2p}$) of $\mathscr{P}^p_{t,Y}$ in $c[t,Y,p]$, which will be computed using representative sets. Moreover, as we have a structured form of a tree decomposition (nice clique-tree) of H, we compute the entries of the table based on the entries of its children, which will be given by recursive formulae. For leaf nodes, which form base cases for recursive formulae, we compute all entries directly.

Next, we give (recursive) formulae for the computation of the table entries. Consider $t \in V(T)$, $Y \in \mathscr{X}_t$ and $p \in [0,k]$. We compute the entry $c[t,Y,k]$ based on the following

cases.

Leaf node: t is a leaf node. In this case, we have $X_t = \emptyset$, and hence $\mathscr{X}_t = \{\emptyset\}$. If $p = 0$, then $\mathscr{P}^p_{t,\emptyset} = \{\emptyset\}$, and $\mathscr{P}^p_{t,\emptyset} = \emptyset$, otherwise. Since, $\mathscr{P}^p_{t,\emptyset}$ is a valid candidate for $c[t,Y,p]$, we set $c[t,Y,p] = \mathscr{P}^p_{t,\emptyset}$. Note that $c[t,Y,p]$ has size at most $1 \leq \binom{2k}{2p}$, and we can compute $c[t,Y,p]$ in polynomial time.

Introduce node: Suppose t is an introduce node with child t' such that $X_t = X_{t'} \cup \{e\}$, where $e \notin X_{t'}$. If $Y \neq \emptyset$ and $p < 1$, then we set $c[t,Y,p] = \emptyset$. Otherwise, we compute the entry as described below. Before computing the entry $c[t,Y,p]$, we first compute a set $\widetilde{\mathscr{P}}^p_{t,Y}$ as follows.

$$\widetilde{\mathscr{P}}^p_{t,Y} = \begin{cases} c[t',Y,p] & \text{if } Y \neq \{e\}; \\ c[t',\emptyset,p-1] \star \{\{e,e^c\}\} & \text{otherwise.} \end{cases} \tag{6.1}$$

Next, we compute $\widehat{\mathscr{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \widetilde{\mathscr{P}}^p_{t,Y}$ of size $\binom{2k}{2p}$, using Theorem 6.2.1. Finally, we set $c[t,Y,p] = \widehat{\mathscr{P}}^p_{t,Y}$.

Correctness: To show the correctness, it is enough to show that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$. If $Y \neq \emptyset$ and $p < 1$, then we correctly set $c[t,Y,p] = \emptyset$. Hereafter, we assume that $Y \neq \emptyset$ then $p \geq 1$. If $Y \neq \{e\}$, then the claim follows from the fact that $c[t,Y,p] = c[t',Y,p]$ and $\mathscr{P}^p_{t,Y} = \mathscr{P}^p_{t',Y}$. Therefore, we consider the case when $Y = \{e\}$. In this case, we observe the following towards proving the claim.

1. $\mathscr{P}^p_{t,Y} = \mathscr{P}^{p-1}_{t',\emptyset} \star \{\{e,e^c\}\}$.
2. $c[t',\emptyset,p-1] \subseteq^{2k-2(p-1)}_{\mathsf{rep}} \mathscr{P}^{p-1}_{t',\emptyset}$.

From item 1 and 2, and Lemma 34, it follows that $c[t',\emptyset,p-1] \star \{\{e,e^c\}\} \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$. This together with Proposition 19, and the fact that $\widehat{\mathscr{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} c[t',\emptyset,p-1] \star \{\{e,e^c\}\}$

implies that $c[t,Y,p] = \widehat{\mathscr{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$.

Forget node: Suppose t is a forget node with child t' such that $X_t = X_{t'} \setminus \{e\}$, where $e \in X_{t'}$. Before computing the entry $c[t,Y,p]$, we first compute a set $\widetilde{\mathscr{P}}^p_{t,Y}$ as follows.

$$\widetilde{\mathscr{P}}^p_{t,Y} = \begin{cases} c[t',Y,p] & \text{if } Y \neq \emptyset; \\ c[t',\emptyset,p] \cup c[t',\{e\},p] & \text{otherwise.} \end{cases} \tag{6.2}$$

Next, we compute $\widehat{\mathscr{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \widetilde{\mathscr{P}}^p_{t,Y}$ of size $\binom{2k}{2p}$, using Theorem 6.2.1. Finally, we set $c[t,Y,p] = \widehat{\mathscr{P}}^p_{t,Y}$.

Correctness: To show the correctness, it is enough to show that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$. If $Y \neq \emptyset$, then the claim follows from the fact that $c[t,Y,p] = c[t',Y,p]$, and $\mathscr{P}^p_{t,Y} = \mathscr{P}^p_{t',Y}$. Therefore, we consider the case when $Y = \emptyset$. In this case, we observe the following towards proving the claim.

1. $c[t',\emptyset,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t',\emptyset}$.
2. $c[t',\{e\},p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t',\{e\}}$.
3. $\mathscr{P}^p_{t,Y} = \mathscr{P}^p_{t',\emptyset} \cup \mathscr{P}^p_{t',\{e\}}$.

From item 1 to 3, and Proposition 20, it follows that $c[t',\emptyset,p] \cup c[t',\{e\},p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$. This together with Proposition 19, and the fact that $\widehat{\mathscr{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} c[t',\emptyset,p] \cup c[t',\{e\},p]$ implies that $c[t,Y,p] = \widehat{\mathscr{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$.

Join node: Suppose t is a join node with children t_1 and t_2, such that $X_t = X_{t_1} = X_{t_2}$. If $Y \neq \emptyset$ and $p < 1$, then we set $c[t,Y,p] = \emptyset$. Otherwise, we compute the entry as described below. Before computing the entry $c[t,Y,p]$, we first compute a set $\widetilde{\mathscr{P}}^p_{t,Y}$ as follows.

$$\tilde{\mathscr{P}}^{p}_{t,Y} = \begin{cases} \bigcup\limits_{i\in[0,p]} (c[t_1,\emptyset,i] \star c[t_2,\emptyset,p-i]) & \text{if } Y=\emptyset; \\ \bigcup\limits_{i\in[p]} (c[t_1,Y,i] \star c[t_2,\emptyset,p-i]) & \text{otherwise.} \end{cases} \tag{6.3}$$

Next, we compute $\widehat{\mathscr{P}}^{p}_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \widetilde{\mathscr{P}}^{p}_{t,Y}$ of size $\binom{2k}{2p}$, using Theorem 6.2.1. Finally, we set $c[t,Y,p] = \widehat{\mathscr{P}}^{p}_{t,Y}$.

Correctness: To show the correctness, it is enough to show that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^{p}_{t,Y}$. If $Y \neq \emptyset$ and $p < 1$, then we correctly set $c[t,Y,p] = \emptyset$. Hereafter, we assume that whenever $Y \neq \emptyset$, we have $p \geq 1$. Next, we consider the following cases depending on whether or not $Y = \emptyset$.

- $Y = \emptyset$. In this case, we have $\mathscr{P}^{p}_{t,Y} = \cup_{i\in[0,p]}(\mathscr{P}^{i}_{t_1,\emptyset} \star \mathscr{P}^{p-i}_{t_1,\emptyset})$. Moreover, for $i \in [0,p]$, we have that $c[t_1,\emptyset,i] \subseteq^{2k-2i}_{\mathsf{rep}} \mathscr{P}^{i}_{t_1,\emptyset}$ and $c[t_2,\emptyset,p-i] \subseteq^{2k-2(p-i)}_{\mathsf{rep}} \mathscr{P}^{p-i}_{t_1,\emptyset}$. Above arguments together with Proposition 20 and Lemma 34 implies that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^{p}_{t,Y}$.

- $Y \neq \emptyset$. In this case, we start by arguing that $\widehat{\mathscr{P}}^{p}_{t,Y} = \cup_{i\in[p]}(c[t_1,Y,i] \star c[t_2,\emptyset,p-i]) \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^{p}_{t,Y}$. To this end, consider a set $A \in \mathscr{P}^{p}_{t,Y}$ of size $2p$ and a set $B \subseteq E \cup E^c$ of size $2k-2p$ such that $A \cup B \in \mathscr{I}$ and $A \cap B = \emptyset$. Observe that by construction of $\mathscr{P}^{p}_{t,Y}$, $A \subseteq V(H_t) \cup (V(H_t))^c$, $A \cap X_t = Y$. Let $A_1 = A \cap V(H_{t_1})$, $A_2 = A \setminus A_1$, and $i^* = |A_1|$. Since $A \in \mathscr{P}^{p}_{t,Y}$, and $\mathscr{P}^{p}_{t,Y}$ is a paired-family, it holds that $A_1^c \cup A_2^c \subseteq A$. Let $B_2 = B \cup A_1 \cup A_1^c$, and note that $|B_2| = 2k-2(p-i^*)$. Moreover, $c[t_2,\emptyset,p-i^*] \subseteq^{2k-2(p-i^*)}_{\mathsf{rep}} \mathscr{P}^{p-i^*}_{t_2,\emptyset}$, and therefore, there is $\tilde{A}_2 \cup \tilde{A}_2^c \in c[t_2,\emptyset,p-i^*]$ such that $(\tilde{A}_2 \cup \tilde{A}_2^c) \cup B_2 \in \mathscr{I}$ and $(\tilde{A}_2 \cup \tilde{A}_2^c) \cap B_2 = \emptyset$. Next, consider $B_1 = B \cup (\tilde{A}_2 \cup \tilde{A}_2^c)$, and note that $|B_1| = 2k-2p+2(p-i^*) = 2k-2i^*$. Since, $c[t_1,Y,i^*] \subseteq^{2k-2i^*}_{\mathsf{rep}} \mathscr{P}^{i^*}_{t_1,Y}$, therefore, there is $\tilde{A}_1 \cup \tilde{A}_1^c \in c[t_1,T,i^*]$ such that $B_1 \cup (\tilde{A}_1 \cup \tilde{A}_1^c) \in \mathscr{I}$ and $B_1 \cap (\tilde{A}_1 \cup \tilde{A}_1^c) = \emptyset$. The above arguments imply that $(\tilde{A}_1 \cup \tilde{A}_1^c) \cup (\tilde{A}_2 \cup \tilde{A}_2^c) \in \mathscr{I}$ and $(\tilde{A}_1 \cup \tilde{A}_1^c) \cap (\tilde{A}_2 \cup \tilde{A}_2^c) =$

$\emptyset$. Hence, by definition of the convolution operation $(\star)$, we have $(\tilde{A}_1 \cup \tilde{A}_1^c) \cup (\tilde{A}_2 \cup \tilde{A}_2^c) \in c[t_1, Y, i^*] \star c[t_2, \emptyset, p - i^*]$. Moreover, $B \cup (\tilde{A}_1 \cup \tilde{A}_1^c) \cup (\tilde{A}_2 \cup \tilde{A}_2^c) \in \mathscr{I}$ and $B \cap (\tilde{A}_1 \cup \tilde{A}_1^c) \cup (\tilde{A}_2 \cup \tilde{A}_2^c) = \emptyset$. Therefore, $\cup_{i \in [p]} (c[t_1, Y, i] \star c[t_2, \emptyset, p - i]) \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$. This together with Proposition 19 implies that $c[t, Y, p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathscr{P}^p_{t,Y}$.

This completes the description of the (recursive) formulae and their correctness for computing all entries of the table. The correctness of the algorithm follows from the correctness of the (recursive) formulae, and the fact that (G, H, k) is a yes-instance of CCBM if and only if $c[r, \emptyset, k] \neq \emptyset$. Next, we move to the running time analysis of the algorithm.

Lemma 37. *The algorithm presented for* CCBM *runs in time* $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$*, where n is the number of vertices in G.*

Proof. We do the running time analysis based on time required to compute an entry $c[t, Y, k]$, for $t \in V(T)$, $Y \in \mathscr{X}_t$ and $p \in [0, k]$. We consider the following cases.

Leaf node: For leaf nodes the entries $c[t, Y, k]$ can be computed in polynomial time.

Introduce node: The algorithm first computes a family $\widetilde{\mathscr{P}}^p_{Y,t}$ from Equation 6.1, which takes $2^{2k} n^{\mathscr{O}(1)}$ time (using Proposition 21). Moreover, $|\widetilde{\mathscr{P}}^p_{Y,t}| \leq \max\{\binom{2k}{2p}, \binom{2k}{2(p-1)}\}$. The algorithm then computes $\hat{\mathscr{P}}^p_{Y,t} \subseteq^{2k-2p}_{rep} \widetilde{\mathscr{P}}^p_{Y,t}$ using Theorem 6.2.1, which takes time bounded by $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$.

Forget node: The algorithm first computes a family $\widetilde{\mathscr{P}}^p_{Y,t}$ from Equation 6.1, which takes at most $\binom{2k}{2p}$ time by standard set union operation. Moreover, $|\widetilde{\mathscr{P}}^p_{Y,t}| \leq 2\binom{2k}{2p}$. The algorithm then computes $\hat{\mathscr{P}}^p_{Y,t} \subseteq^{2k-2p}_{rep} \widetilde{\mathscr{P}}^p_{Y,t}$. This takes time $|\tilde{\mathscr{P}}^p_{Y,t}| \binom{2k}{2p}^{\omega-1} n^{\mathscr{O}(1)} \leq \binom{2k}{2p}^{\omega} n^{\mathscr{O}(1)} \leq 4^{\omega k} n^{\mathscr{O}(1)}$. Therefore, the time required to compute an entry at forget node is at most $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$.

Join node: The algorithm first computes a family $\widetilde{\mathscr{P}}^p_{Y,t}$ from Equation 6.3, which takes at most $2^{2k} n^{\mathscr{O}(1)}$ time by Proposition 21 and standard set union operation. Moreover,

$|\widetilde{\mathscr{P}}^p_{Y,t}| \leq 2^{\mathscr{O}(\omega k)}$. Now the algorithm applies Theorem 6.2.1 on $\tilde{\mathscr{P}}^p_{Y,t}$ and computes $\hat{\mathscr{P}}^p_{Y,t} \subseteq^{2k-2p}_{rep} \tilde{\mathscr{P}}^p_{Y,t}$. This takes time bounded by $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$. Therefore, the time required to compute an entry at join node is at most $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$.

The time to compute an entry $c[t,Y,k]$ is at most $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$. Moreover, the number of entries is bounded by $|V(T)| \cdot k \cdot n \in n^{\mathscr{O}(1)}$. Thus, the running time of the algorithm is bounded by $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$. □

Theorem 6.4.1. CCBM *admits an* FPT *algorithm running in time* $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$.

6.4.2 FPT algorithm for CHORDAL CONFLICT MATCHING.

We design an FPT algorithm for CHORDAL CONFLICT MATCHING, using the algorithm for CCBM (Theorem 6.4.1). Let (G,H,k) be an instance of CF-MM, where H is a chordal graph and G is a graph on n vertices. We assume that G is a graph on vertex set $[n]$, which can easily be achieved by renaming vertices.

The algorithm starts by computing an $(n,2k)$-universal set $\mathscr{F}$, using Proposition 22. For each set $A \in \mathscr{F}$, the algorithm constructs an instance $I_A = (G_A, L_A, R_A, H_A, k)$ of CCBM as follows. We have $V(G_A) = V(G)$, $L_A = A$, $R = V(G) \setminus A$, $E(G_A) = \{uv \in E(G) \mid u \in L_A, v \in R_A\}$, and $H_A = H[E(G_A)]$. Note that H_A is a chordal graph because chordal graphs are closed under induced subgraphs and disjoint unions. The algorithm decides the instance I_A using Theorem 6.4.1, for each $A \in \mathscr{F}$. The algorithm outputs yes if and only if there is $A \in \mathscr{F}$, such that I_A is a yes-instance of CCBM.

Theorem 6.4.2. *The algorithm presented for* CF-MM *is correct, Moreover, it runs in time* $2^{\mathscr{O}(\omega k)} k^{\mathscr{O}(\log k)} n^{\mathscr{O}(1)}$, *where* $\omega < 2.373$ *is the exponent of matrix multiplication and n is the number of vertices in the input graph.*

Proof. Let (G,H,k) be an instance of CF-MM, where H is a chordal graph and G is a graph on vertex set $[n]$. Clearly, if the algorithm outputs yes, then indeed (G,H,k) is a

yes-instance of CF-MM. Next, we argue that if (G,H,k) is a yes-instance of CF-MM then the algorithm returns yes. Suppose there is a solution $M \subseteq E(G)$ to CF-MM in (G,H,k). Let $S = \{i, j \mid ij \in M\}$, and $L = \{i \mid \text{there is } j \in [n] \text{ such that } ij \in M \text{ and } i < j\}$. Observe that $|S| = 2k$. Since $\mathscr{F}$ is an $(n,2k)$-universal set, there is $A \in \mathscr{F}$ such that $A \cap S = L$. Note that S is a solution to CCBM in I_A. This together with Theorem 6.4.1 implies that the algorithm will return yes as output.

Next, we prove the claimed running time of the algorithm. The algorithm computes $(n,2k)$-universal set of size $\mathscr{O}(2^{2k}k^{\mathscr{O}(\log k)}\log n)$, in time $\mathscr{O}(2^{2k}k^{\mathscr{O}(\log k)}\ n\log n)$, using Proposition 22. Then for each $A \in \mathscr{F}$, the algorithm creates an instance I_A of CCBM in polynomial time. Furthermore, it resolves the I_A of CCBM in time $\mathscr{O}(2^{\mathscr{O}(\omega k)}n^{\mathscr{O}(1)})$ using Theorem 6.4.1. Hence, the running time of the algorithm is bounded by $2^{\mathscr{O}(\omega k)}k^{\mathscr{O}(\log k)}n^{\mathscr{O}(1)}$.

□

6.5 FPT algorithms for CF-MM and CF-SP with matroid constraints

In this section, we study the problems CF-MM and CF-SP, where the conflicting condition is being an independent set in a (representable) matroid. Due to technical reasons (which will be clear later) for the above variant of CF-MM, we will only consider the case when the matroid is repsesentable over $\mathbb{Q}$ (the field of rationals).

6.5.1 FPT algorithm for MATROID CF-MM

We study a variant of the problem CF-MM, where the conflicting condition is being an independent set in a matroid representable over $\mathbb{Q}$. We call this variant of CF-MM as

RATIONAL MATROID CF-MM (RAT MAT CF-MM, for short), which is formally defined below.

RATIONAL MATROID CF-MM (RAT MAT CF-MM) **Parameter:** k
Input: A graph G, a matrix $A_{\mathscr{M}}$ (representing a matroid $\mathscr{M}$ over $\mathbb{Q}$) with columns indexed by $E(G)$, and an integer k.
Question: Is there a matching $M \subseteq E(G)$ of size at most k, such that the set of columns in M are linearly independent (over $\mathbb{Q}$)?

We design an FPT algorithm for RAT MAT CF-MM. Towards designing an algorithm for RAT MAT CF-MM, we first give an FPT algorithm for a restricted version of RAT MAT CF-MM, where the graph in which we want to compute a matching is a bipartite graph. We call this variant of RAT MAT CF-MM as RAT MAT CF-BIPARTITE MATCHING (RAT MAT CF-BM). We then employ the algorithm for RAT MAT CF-BM to design an FPT algorithm for RAT MAT CF-MM.

6.5.1.1 FPT algorithm for RAT MAT CF-BM

We design an FPT algorithm for the problem RAT MAT CF-BM, where the conflicting condition is being an independent set in a matroid (representable over $\mathbb{Q}$) and the graph where we want to compute a matching is a bipartite graph. This problem is formally defined below.

RAT MAT CF-BIPARTITE MATCHING (RAT MAT CF-BM)
Input: A bipartite graph $G = (V,E)$ with vertex bipartition L,R, a matrix $A_{\mathscr{M}}$ (representing a matroid $\mathscr{M}$ over $\mathbb{Q}$) with columns indexed by E, and an integer k.
Parameter: k
Question: Is there a matching $M \subseteq E$ of size k in G, such that the set of columns in M are linearly independent (over $\mathbb{Q}$)?

Our algorithm takes an instance of RAT MAT CF-BM and generates an instance of 3-MATROID INTERSECTION, and then employs the known algorithm for 3-MATROID

INTERSECTION to resolve the instance. In the following, we formally define the problem 3-MATROID INTERSECTION.

3-MATROID INTERSECTION	**Parameter:** k
Input: Matrices $A_{\mathscr{M}_1}, A_{\mathscr{M}_2}$, and $A_{\mathscr{M}_3}$ over $\mathbb{F}$ (representing matroids $\mathscr{M}_1, \mathscr{M}_2$, and $\mathscr{M}_3$, respectively, on the same ground set E) with columns indexed by E, and an integer k.	
Question: Is there a set $M \subseteq E$ of size k, such that M is independent in each $\mathscr{M}_i$, for $i \in [3]$?	

Before moving further, we briefly explain why we needed an additional constraint that the input matrix is representable over $\mathbb{Q}$. Firstly, we will be using partition matroids which are representable only on fields of large enough size, and we want all the matroids, that is, the one which is part of the input and the ones that we create, to be representable over the same field. Secondly, the algorithmic result (with the desired running time) we use for 3-MATROID INTERSECTION works only for certain types of fields.

Next, we state an algorithmic result regarding 3-MATROID INTERSECTION [79], which is be used by the algorithm. We note that we only state a restricted form of the algorithmic result for 3-MATROID INTERSECTION in [79], which is enough for our purpose.

Proposition 24. [Proposition 4.8 [79] (partial)] 3-MATROID INTERSECTION *can be solved in* $\mathscr{O}(2^{3\omega k}\|A_M\|^{\mathscr{O}(1)})$ *time, when the matroids are represented over* $\mathbb{Q}$.

We are now ready to prove the desired result.

Theorem 6.5.1. RAT MAT CF-BM *can be solved in* $\mathscr{O}(2^{3\omega k}\|A_M\|^{\mathscr{O}(1)})$ *time.*

Proof. Let $(G=(V,E), L, R, A_{\mathscr{M}}, k)$ be an instance of RAT MAT CF-BM, where the matrix $A_{\mathscr{M}}$ represents a matroid $\mathscr{M}=(E,\mathscr{I})$ over $\mathbb{Q}$. Let $\mathscr{M}_L=(E,\mathscr{I}_L), \mathscr{M}_R=(E,\mathscr{I}_R)$ be the partition matroids as defined in Section 6.4. Next we compute matrix representations $A_{\mathscr{M}_L}$ and $A_{\mathscr{M}_R}$ of matroids $\mathscr{M}_L, \mathscr{M}_R$, respectively, using Proposition 17. Now, we solve 3-MATROID INTERSECTION on the instance $(\mathscr{M}, A_{\mathscr{M}_L}, A_{\mathscr{M}_R}, k)$ (over $\mathbb{Q}$) using Proposition 24, and return the same answer, as returned by the algorithm in it. The correctness

follows directly from the following. $S \subseteq E$ is a matching in G if and only if S is an independent set in $\mathscr{M}_L$ and $\mathscr{M}_R$, that is, $S \in \mathscr{I}_L \cap \mathscr{I}_R$. The claimed running time follows from Proposition 17 and Proposition 24. □

6.5.1.2 FPT algorithm for RAT MAT CF-MM

We design an FPT algorithm for RAT MAT CF-MM, using the algorithm for RAT MAT CF-BM (Theorem 6.4.1). Let $(G, A_{\mathscr{M}}, k)$ be an instance of RAT MAT CF-MM, where the matrix $A_{\mathscr{M}}$ represents a matroid $\mathscr{M} = (E, \mathscr{I})$ over $\mathbb{Q}$. We assume that G is a graph with the vertex set $[n]$, which can easily be achieved by renaming vertices.

The algorithm starts by computing an $(n, 2k)$-universal set $\mathscr{F}$, using Proposition 22. For each set $X \in \mathscr{F}$, the algorithm constructs an instance $I_X = (G_X, L_X, R_X, A_{\mathscr{M}}, k)$ of RAT MAT CF-BM as follows. We have $V(G_X) = V(G)$, $L_X = X$, $R = V(G) \setminus X$, $E(G_X) = \{uv \in E(G) \mid u \in L_X, v \in R_X\}$. The algorithm decides the instance I_X using Theorem 6.5.1, for each $X \in \mathscr{F}$. The algorithm outputs yes if and only if there is $X \in \mathscr{F}$, such that I_X is a yes-instance of RAT MAT CF-BM.

Theorem 6.5.2. *The algorithm presented for* RAT MAT CF-MM *is correct. Moreover, it runs in time* $\mathscr{O}(2^{(3\omega+2)k} k^{\mathscr{O}(\log k)} \|A_M\|^{\mathscr{O}(1)} n^{\mathscr{O}(1)})$.

Proof. Let $(G, A_{\mathscr{M}}, k)$ be an instance of RAT MAT CF-MM, where matrix $A_{\mathscr{M}}$ represent a matroid $\mathscr{M} = (E, \mathscr{I})$ over field $\mathbb{F}$. Clearly, if the algorithm outputs yes, then indeed $(G, A_{\mathscr{M}}, k)$ is a yes-instance of RAT MAT CF-MM. Next, we argue that if $(G, A_{\mathscr{M}}, k)$ is a yes-instance of RAT MAT CF-MM then the algorithm returns yes. Suppose there is a solution $M \subseteq E(G)$ to RAT MAT CF-MM in $(G, A_{\mathscr{M}}, k)$. Let $S = \{i, j \mid ij \in M\}$, and $L = \{i \mid \text{there is } j \in [n] \text{ such that } ij \in M \text{ and } i < j\}$. Observe that $|S| = 2k$. Since $\mathscr{F}$ is an $(n, 2k)$-universal set, there is $X \in \mathscr{F}$ such that $X \cap S = L$. Note that S is a solution to RAT MAT CF-BM in I_X. This together with Theorem 6.5.1 implies that the algorithm will return yes as the output.

Next, we prove the claimed running time of the algorithm. The algorithm computes $(n,2k)$-universal set of size $\mathcal{O}(2^{2k}k^{\mathcal{O}(\log k)}\log n)$, in time $\mathcal{O}(2^{2k}k^{\mathcal{O}(\log k)}\ n\log n)$, using Proposition 22. Then for each $X \in \mathscr{F}$, the algorithm creates an instance I_X of RAT MAT CF-BM in polynomial time. Furthermore, it resolves the I_X of RAT MAT CF-BM in time $\mathcal{O}(2^{3\omega k}\|A_M\|^{\mathcal{O}(1)})$ using Theorem 6.5.1. Hence, the running time of the algorithm is bounded by $\mathcal{O}(2^{(3\omega+2)k}k^{\mathcal{O}(\log k)}\|A_M\|^{\mathcal{O}(1)}n^{\mathcal{O}(1)})$.

□

6.5.2 FPT algorithm for MATROID CF-SP

In this section, we design an FPT algorithm for MATROID CF-SP. In the following, we formally define the problem MATROID CF-SP.

MATROID CF-SP **Parameter:** k
Input: A graph G, (distinct) vertices $s,t \in V(G)$, a matrix $A_{\mathscr{M}}$ (representing a matroid $\mathscr{M}$, over a field $\mathbb{F}$) with columns indexed by $E(G)$, and an integer k.
Question: Is there a set $M \subseteq E(G)$ of size at most k, such that there is an st-path in $G[M]$ and the set of columns in M are linearly independent (over $\mathbb{F}$)?

Our algorithm is based on a dynamic programming over representative families. Let $(G,s,t,A_{\mathscr{M}},k)$ be an instance of MATROID CF-SP. Before moving to the description of the algorithm, we need to define some notations.

For distinct vertices $u,v \in V(G)$ and an integer p, we define the following.

$$\mathscr{P}^p_{uv} = \{X \subseteq E(G) \mid |X| = p, \text{ there is a } uv\text{-path in } G[X] \text{ containing all edges in } X, \text{ and } X \in \mathscr{I}\} \tag{6.4}$$

By the definition of convolution of sets, it is easy to see that

$$\mathscr{P}^p_{uv} = \bigcup_{wv \in E(G)} \mathscr{P}^{p-1}_{uw} \star \{\{wv\}\}$$

Now we are ready to describe our algorithm Alg-Mat-CF-SP for MATROID CF-SP. We aim to store, for each $v \in V(G) \setminus \{s\}$, $p \leq k$, and $q \leq k-p$, a q-representative set $\widehat{\mathscr{P}}^{pq}_{sv}$, of $\mathscr{P}^p_{sv}$, of size $\binom{p+q}{q}$. Notice that for each $v \in V(G) \setminus \{s\}$, we can compute $\mathscr{P}^1_{sv}$ in polynomial time, since $\mathscr{P}^1_{sv} = \{sv\}$ if $sv \in E(G)$, and is empty otherwise. Moreover, since $|\mathscr{P}^1_{sv}| \leq 1$, therefore, we can set $\widehat{\mathscr{P}}^{1q}_{sv} = \mathscr{P}^1_{sv}$, for each $q \leq k-1$. Next, we iteratively compute, for each $p \in \{2,3,\cdots,k\}$, in increasing order, for each $q \leq k-p$, a q-representative $\widehat{\mathscr{P}}^{pq}_{sv}$, of $\mathscr{P}^p_{sv}$. The algorithm Alg-Mat-CF-SP is given in Algorithm 3.

Next, we prove a lemma which will be useful in establishing the correctness of Alg-Mat-CF-SP.

Lemma 38. *For each $p \in [k]$, $q \in [0,k-p]$, and $v \in V(G) \setminus \{s\}$, the family $\widehat{\mathscr{P}}^{pq}_{sv}$ computed by* Alg-Mat-CF-SP *is a q-representative of $\mathscr{P}^p_{sv}$, and is of size at most $\binom{p+q}{q}$. Moreover, the algorithm computes all sets in $\{\widehat{\mathscr{P}}^{pq}_{sv} \mid p \in [k], q \in [0,k-p], v \in V(G) \setminus \{s\}\}$ in time $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$.*

Proof. We prove the claim by induction on p. Consider $v \in V(G) \setminus \{s\}$. For $p = 1$, the set $\mathscr{P}^1_{sv} = \{sv\}$ if $sv \in E(G)$, and is empty otherwise. Moreover, for each $q \in [0,k-1]$, Alg-Mat-CF-SP sets $\widehat{\mathscr{P}}^{1q}_{sv} = \mathscr{P}^1_{sv}$. Hence, for each $q \in [0,k-1]$, we have $\widehat{\mathscr{P}}^{1q}_{sv} \subseteq^q_{\mathsf{rep}} \mathscr{P}^1_{sv}$. Moreover, the set $\widehat{\mathscr{P}}^{1q}_{sv}$ is computed by the algorithm in polynomial time.

For induction hypothesis, we assume that for $t \in \mathbb{N}_{\geq 1}$, for each $p \leq t$, $q \in [0,k-p]$, and $v \in V(G) \setminus \{s\}$, we have $\widehat{\mathscr{P}}^{pq}_{sv} \subseteq^q_{\mathsf{rep}} \mathscr{P}^p_{sv}$. Next, consider $p = t+1$, $q \in [0,k-(t+1)]$, and $v \in V(G) \setminus \{s\}$. The step of the algorithm, where it computes $\widehat{\mathscr{P}}^{(t+1)q}_{sv}$, it has already computed (correctly), for each $p \leq t$, $q \in [0,k-p]$, and $u \in V(G) \setminus \{s\}$, the set $\widehat{\mathscr{P}}^{pq}_{su} \subseteq^q_{\mathsf{rep}} \mathscr{P}^p_{su}$. This follows from the iteration of the algorithm over p from 1 to k in

Algorithm 3 Alg-Mat-CF-SP

Input: A graph G, (distinct) vertices $s,t \in V(G)$, a matrix $A_{\mathscr{M}}$ (over $\mathbb{F}$), and an integer k.
Output: If there is $M \subseteq E(G)$ of size at most k, such that there is an $s-t$ path in $G[M]$ and the set of columns in M are linearly independent (over $\mathbb{F}$) then yes. Otherwise, no.

```
for each v ∈ V(G) \ {s} do
    if sv ∈ E(G) then
        return 𝒫^1_sv = {sv}
    else
        𝒫^1_sv = ∅
    end if
    for q = 0 to k − 1 do
        Set 𝒫̂^{1q}_sv = 𝒫^1_sv
    end for
end for
for p = 2 to k do
    for q = 0 to k − p do
        for each v ∈ V(G) \ {s} do
            Let 𝒫̃^{pq}_sv = ∪_{wv∈E(G)} 𝒫̂^{(p−1)(q+1)}_sw ⋆ {{wv}}
            Compute 𝒫̂^{pq}_sv ⊆^{k−p}_rep 𝒫̃^{pq}_sv using Theorem 6.2.1
        end for
    end for
end for
for p = 1 to k do
    for q = 0 to k − p do
        if 𝒫̂^{pq}_st ≠ ∅ then
            return yes
        end if
    end for
end for
return no
```

increasing order at Step 6 (and Step 1). The algorithm sets $\widetilde{\mathscr{P}}_{sv}^{(t+1)q} = \cup_{wv \in E(G)} \widehat{\mathscr{P}}_{sw}^{(t)(q+1)} \star \{\{wv\}\}$, and then sets $\widehat{\mathscr{P}}_{sv}^{(t+1)q}$ to be the q-representative set of $\widetilde{\mathscr{P}}_{sv}^{(t+1)q}$ returned by Theorem 6.2.1, which is of size at most $\binom{t+1+q}{t+1}$. If we show that $\widetilde{\mathscr{P}}_{sv}^{(t+1)q} \subseteq^{q}_{\mathsf{rep}} \mathscr{P}_{sv}^{t+1}$, then by Proposition 19 it will follow that $\widehat{\mathscr{P}}_{sv}^{(t+1)q} \subseteq^{q}_{\mathsf{rep}} \mathscr{P}_{sv}^{t+1}$. But $\widetilde{\mathscr{P}}_{sv}^{(t+1)q} \subseteq^{q}_{\mathsf{rep}} \mathscr{P}_{sv}^{t+1}$ follows from Lemma 34 and Proposition 20. Also, note that each entry can be computed in time $\mathscr{O}(2^{\mathscr{O}(\omega k)} n^{\mathscr{O}(1)})$. □

Using Lemma 38, we obtain the following theorem.

Theorem 6.5.3. *The algorithm* Alg-Mat-CF-SP *is correct. Moreover, it runs in time*

$\mathcal{O}(2^{\mathcal{O}(\omega k)}n^{\mathcal{O}(1)})$.

Proof. Let $(G,s,t,A_{\mathscr{M}},k)$ be an instance of MATROID CF-SP. We claim that $(G,s,t,A_{\mathscr{M}},k)$ is a yes-instance of MATROID CF-SP if and only if Alg-Mat-CF-SP outputs yes. In the forward direction, let $(G,s,t,A_{\mathscr{M}},k)$ be a yes-instance of MATROID CF-SP. Since, using Lemma 38, Alg-Mat-CF-SP computes a q-representative of $\mathscr{P}^p_{sv}$ of size at most $\binom{p+q}{q}$, for each $p \in [k]$, $q \in [0,k-p]$, and $v \in V(G) \setminus \{s\}$, therefore, the algorithm also computes a q-representative family for $\mathscr{P}^k_{st}$. By the definition of representative set and construction of our family $\mathscr{P}^k_{st}$, $\widehat{\mathscr{P}}^k_{st}$ also contains a $s-t$ path and hence, the algorithm outputs yes. In the reverse direction, if the algorithm outputs yes then by construction of family $\widehat{\mathscr{P}}^k_{st}$, if $P \in \widehat{\mathscr{P}}^k_{st}$, then it is a conflict-free $s-t$ path in G. This completes the correctness of our algorithm. Moreover, the running time bound of the algorithm follows from Lemma 38. □

Theorem 6.5.3 will also result into an FPT algorithm for CF-SP when the conflict graph is a cluster graph.

Corollary 8. CONFLICT FREE SHORTEST PATH *parameterized by the solution size is* FPT*, when the conflict graph is a cluster graph.*

Proof. Let (G,H,k) be an instance of CF-SP, where H is a cluster graph. We construct a partition matroid, $\mathscr{M}_H = (U,\mathscr{I})$, corresponding to graph H as follows. We define ground set as $U = V(H)$. Now, partition U as $U_i = V(C_i)$, for each clique C_i in H and $a_i = 1$, for $U_i \in U$. By the construction of $\mathscr{M}_H$, we have for $S \subseteq V(H)$, S is an independent set in H if and only if S is independent in $\mathscr{M}_H$. Next, we compute a matrix M representing $\mathscr{M}_H$, using Proposition 17 in polynomial time.Now we use Alg-Mat-CF-SP with input (G,M,k), and return the same output. The correctness of our algorithm follows from correctness of the algorithm Alg-Mat-CF-SP (Theorem 6.5.3), and by construction of the instance (G,M,k). Note that the matrix M representing $\mathscr{M}_H$ can be computed in polynomial time.

This together with Theorem 6.5.3 implies the claimed running time bound, This concludes the proof. □

6.6 FPT Algorithm for d-degenerate Conflict Graphs

In this section, we show that CF-MM and CF-SP both are in FPT, when the conflict graph H is a d-degenerate graphs. These algorithms are based on the notion of independence covering family, which was introduced in [81].

Before moving onto description of our algorithms, we recall the notion of independence covering family.

Definition 27 ([81])**.** For a graph $H^\star$ and an integer k, a *k-independence covering family*, $\mathscr{I}(H^\star, k)$, is a family of independent sets in $H^\star$ such that for any independent set I' in $H^\star$ of size at most k, there is a set $I \in \mathscr{I}(H^\star, k)$ such that $I' \subseteq I$.

Our algorithms rely on the construction of k-independence covering family, for a family of graphs. But before dwelling into these details, we first design an algorithm for an annotated version of the CF-MM and CF-SP problems, which we call ANNOTATED CF-MM and ANNOTATED CF-SP, respectively. In the ANNOTATED CF-MM (ANNOTATED CF-SP) problem, the input to CF-MM (CF-SP) is annotated with a k-independence covering family.

6.6.1 Algorithms for ANNOTATED CF-MM and ANNOTATED CF-SP

In this section, we study the problems ANNOTATED CF-MM and ANNOTATED CF-SP, which are formally defined below.

Algorithm 4 Alg-CF-MM (Alg-CF-SP)

Input: A graph G,((distinct) vertices $s,t \in V(G)$), a conflict graph H, an integer k, and a k-independence covering family $\mathscr{F}$ of H.

Output: If there a set $M \subseteq E$ of size k in G such that M is a matching in G (there is an $s-t$ path in $G[M]$) and M is an independent set in H, then yes, and no otherwise.

```
for each I ∈ F do
    Let G_I be the graph with V(G_I) = V(G) and E(G_I) = I
    if G_I has a matching (path) of size k then
        return yes
    end if
end for
return no
```

ANNOTATED CF-MM **Parameter:** $k+|\mathscr{F}|$

Input: A graph $G=(V,E)$, a conflict graph $H=(E,E')$, an integer k, and a k-independence covering family $\mathscr{F}$ of H.

Question: Is there a matching $M \subseteq E$ of size k in G such that M is an independent set in H?

ANNOTATED CF-SP **Parameter:** $k+|\mathscr{F}|$

Input: A graph $G=(V,E)$, (distinct) vertices $s,t \in V$, a conflict graph $H=(E,E')$, an integer k, and a k-independence covering family $\mathscr{F}$ of H.

Question: Is there a set $M \subseteq E$ of size at most k, such that there is an $s-t$ path in $G[M]$ and M is an independent set in H?

The algorithm that we design for ANNOTATED CF-MM runs in time polynomial in the size of the input. We give the algorithm Alg-CF-MM (Alg-CF-SP) (Algorithm 4) for ANNOTATED CF-MM (ANNOTATED CF-SP).

In the following lemma we prove the correctness of Alg-CF-MM (Alg-CF-SP).

Lemma 39. *The algorithm* Alg-CF-MM (Alg-CF-SP) *is correct. Moreover, the algorithm runs in time polynomial in the size of the input.*

Proof. Let $(G,(s,t),H,k,\mathscr{F})$ be an instance of ANNOTATED CF-MM (ANNOTATED CF-SP). We show that $(G,(s,t),H,k,\mathscr{F})$ is a yes-instance of ANNOTATED CF-MM

(ANNOTATED CF-SP) if and only if Alg-CF-MM (Alg-CF-SP) outputs yes.

Note that the reverse direction easily follows from the fact that $\mathscr{F}$ is a family of independent sets in H. Therefore, we only need to prove the forward direction. In the forward direction, let $(G,(s,t),H,k,\mathscr{F})$ be a yes-instance of ANNOTATED CF-MM (ANNOTATED CF-SP) and S be one of its solution. Since $\mathscr{F}$ is a k-independence covering family, there is $I \in \mathscr{F}$ such that $S \subseteq I$ (see Definition 27). Hence, in the iteration where the algorithm considers I in its for loop, the graph G_I has S as a matching (there is an $s-t$ path in $G_I[S]$). Therefore, the algorithm outputs yes at this iteration.

The running time analysis follows from the fact that maximum matching (shortest path) can be computed in polynomial time [47, 88]([41, 9]). □

Next, we use Alg-CF-MM (Alg-CF-SP) together with Independence Covering Lemma of [81] to obtain algorithms for CF-MM (CF-SP) when the conflict graph is d-degenerate or nowhere dense graph. Towards this, first we recall some lemmas from [81] that we use in our algorithms.

Proposition 25. [Lemma 1.1,[81]] *There is a randomized algorithm running in polynomial time, that given a d-degenerate graph $H^\star$ and an integer k as input, outputs an independent set I, such that for every independent set I' of size at most k in graph $H^\star$, the probability that $I' \subseteq I$ is at least $(\binom{k(d+1)}{k} \cdot k(d+1))^{-1}$.*

Proposition 26. [Lemmas 3.2 and 3.3,[81]] *There are two deterministic algorithms $\mathscr{A}_1$ and $\mathscr{A}_2$, which given a d-degenerate graph $H^\star$ and an integer k, output independence covering families $\mathscr{I}_1(H^\star,k)$ and $\mathscr{I}_2(H^\star,k)$, respectively, such that the following conditions are satisfied.*

- *$\mathscr{A}_1$ runs in time $\mathscr{O}(|\mathscr{I}_1(H^\star,k)| \cdot (n+m))$, where $|\mathscr{I}_1(H^\star,k)| = \binom{k(d+1)}{k} \cdot 2^{o(k(d+1))} \cdot \log n$.*
- *$\mathscr{A}_2$ runs in time $\mathscr{O}(|\mathscr{I}_2(H^\star,k)| \cdot (n+m))$, where $|\mathscr{I}_2(H^\star,k)| = \binom{k^2(d+1)^2}{k} \cdot (k(d+1))^{\mathscr{O}(1)} \cdot \log n$.*

Next, using Proposition 25 and 26, together with Alg-CF-MM (Alg-CF-SP), we obtain randomized and deterministic algorithms, respectively for CF-MM (CF-SP), when the conflict graph is a d-degenerate graph.

Theorem 6.6.1. *There is a randomized algorithm, which given an instance (G,H,k) of* CF-MM(CF-SP)*, where H is a d-degenerate graph, in time $\binom{k(d+1)}{k} \cdot k(d+1) \cdot n^{\mathcal{O}(1)}$, either reports a failure or correctly outputs that the input is a* yes*-instance of* CF-MM(CF-SP)*. Moreover, if the input is a* yes*-instance of* CF-MM(CF-SP)*, then the algorithm outputs correct answer with a constant probability.*

Proof. Let $(G,(s,t),H,k)$ be an instance CF-MM (CF-SP), where H is a d-degenerate graph.

We repeat the following procedure $(\binom{k(1+d)}{k} \cdot k(d+1))$ many times.

1. The algorithm computes an independent set I in (H,k) using Proposition 25.
2. The algorithm calls Alg-CF-MM (Alg-CF-SP) with input $(G,(s,t)H,k,\{I\})$.

The algorithm outputs yes, if in one of the calls to Alg-CF-MM (Alg-CF-SP), it receives a yes. Otherwise, the algorithm outputs no. The running time analysis of the above procedure follows from Proposition 25 and Lemma 39. Also, given a yes-instance, the guarantee on success probability follows from Proposition 25, the number of repetitions, and Lemma 39. Moreover, from Lemma 39 the yes output returned by the algorithm is indeed the correct output to CF-MM(CF-SP)for the given instance. This concludes the proof.

□

Theorem 6.6.2. CF-MM (CF-SP) *admits a deterministic algorithm running in time* $\min\left\{\binom{k(d+1)}{k} \cdot 2^{o(k(d+1))} \cdot \log n, \binom{k^2(d+1)^2}{k} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot \log n\right\} \cdot n^{\mathcal{O}(1)}$*, when the conflict graph is a d-degenerate graph.*

Proof. Let $(G,(s,t),H,k)$ be an instance CF-MM (CF-SP), where H is a d-degenerate graph. The algorithm starts by computing a k-independence covering family $\mathscr{I}(H,k)$ of H, using Proposition 8. Next, we call Alg-CF-MM (Alg-CF-SP) with the input $(G,(s,t),H,k,$ $\mathscr{I}(H,k))$. The correctness and running time analysis of the above procedure follows from Proposition 8 and Lemma 39. This completes the proof. □

6.7 Conclusion

We studied conflict-free (parameterized) variants of MAXIMUM MATCHING (CF-MM) and SHORTEST PATH (CF-SP). We showed that both CF-MM and CF-SP are W[1]-hard, when parameterized by the solution size. In fact, our W[1]-hardness result for CF-MM holds even when the graph where we want to compute a matching is itself a matching and W[1]-hardness result of CF-SP holds even when the conflict graph is a unit interval graph. Then, we restricted our attention to having conflict graphs belonging to some families of graphs, where the INDEPENDENT SET problem is either polynomial time solvable or solvable in FPT time. In particular, we considered the family of chordal graphs and the family of d-degenerate graphs. For the CF-MM problem, we gave an FPT algorithm, when the conflict graph belongs to the family of chordal graphs. We observed that, we cannot obtain an FPT algorithm for the CF-SP problem when the conflict graph is a chordal graph. This holds because unit-interval graphs are chordal, and the problem CF-SP is W[1]-hard, even when the conflict graph is a unit-interval graph. For conflict graphs being d-degenerate, we obtained FPT algorithms for both CF-MM and CF-SP. Our results hold even when the conflict graph is a nowhere dense graph. Finally, we studied a variant of CF-MM and CF-SP, where instead of conflicting conditions being imposed by independent sets in a conflict graph, they are imposed by independence constraints in a (representable) matroid. We gave FPT algorithms for the above variant of both CF-MM and CF-SP.

An interesting question is to obtain (parameterized) dichotomy results for CF-MM and CF-SP, based on the families of graphs where the input graphs belong to. Another direction could be studying kernelization complexity for different families of graphs, and also to see what all FPT problems remain FPT with the conflicting constraints.

Part III

FVS in Hypergraphs

Chapter 7

Feedback Vertex Set in Hypergraphs

7.1 Introduction

It would be an understatement to say that VERTEX COVER (VC) (given an undirected graph G and a positive integer k, does there exist a set S of k vertices which intersects every edge in G) and FEEDBACK VERTEX SET (FVS) (given an undirected graph G and a positive integer k, does there exist a set S (*called feedback vertex set or in short fvs*) of k vertices which intersects every cycle in G) have played a pivotal role in the development of the field of Parameterized Complexity. While there has been no improvement in the parameterized algorithm for VC in the last 14 years [27] (the conference version appeared in MFCS 2006), faster algorithms for FVS have been developed over the last decade. The best known algorithm for VC runs in time $\mathcal{O}(1.2738^k + kn)$ [27]. On the other hand, for FVS, the first deterministic $\mathcal{O}(c^k n^{\mathcal{O}(1)})$ algorithm was designed only in 2005; independently by Dehne et al. [37] and Guo et al. [59]. It is important to note here that a randomized algorithm for FVS with running time $\mathcal{O}(4^k n^{\mathcal{O}(1)})$ [8] was known in as early as 1999. The deterministic algorithms led to the race of improving the base of the exponent for FVS algorithms and several algorithms [22, 23, 26, 31, 61, 70, 77], both deterministic and randomized, have been designed. Until few months ago the best known deterministic algorithm for FVS

ran in time $3.619^k n^{\mathcal{O}(1)}$ [70], while the Cut and Count technique by Cygan et al. [31] gave the best known randomized algorithm running in time $3^k n^{\mathcal{O}(1)}$. However, just in last few months both these algorithms have been improved; Iwata and Kobayashi [61, IPEC 2019] designed the fastest known deterministic algorithm with running time $\mathcal{O}(3.460^k n)$ and Li and Nederlof [77, SODA 2020] designed the fastest known randomized algorithm with running time $2.7^k n^{\mathcal{O}(1)}$. We would like to remark that many variants of FVS have been studied in literature such as CONNECTED FVS [31, 91], INDEPENDENT FVS [2, 78, 90], SIMULTANEOUS FVS [4, 112] and SUBSET FVS [33, 63, 64, 67, 82].

The main objective of this chapter is a study of FVS on hypergraphs. Recall that hypergraphs are essentially a set family H: we have a universe $V(H)$ and a family of hyperedges $E(H)$, where each hyperedge (or an edge) is a subset of $V(H)$. When every hyperedge in $E(H)$ is of size at most d, it is known as a d-hypergraph. Observe that when each hyperedge is of size *exactly* two, we get an undirected graph. The natural question is, how does VC generalize to hypergraphs. Let (G,k) be an instance of VC. Then, we can view VC as the following problem: Given a hypergraph with vertex set $V(G)$ and the set of hyperedges $E(G)$, does there exist a set of k vertices that intersects every hyperedge. Thus, VC is a special case of HITTING SET (HS): Given a hypergraph H and a positive integer k, does there exist a set of k vertices that intersects every hyperedge. When the size of each hyperedge is upper bounded by d, we refer to the problem as the d-HITTING SET (d-HS) problem. Observe that VC is equivalent to the 2-HS problem. It is well known that HS does not admit an algorithm with running time $f(k)n^{\mathcal{O}(1)}$ for any function f that depends on k alone. That is, the problem is known to be W[2]-hard. On the other hand, d-HS is solvable in time $d^k n^{\mathcal{O}(1)}$ and admits a kernel of size $\mathcal{O}(k^d)$ [1, 49]. It is worth to note that a lower bound of size $\mathcal{O}(k^{d-\varepsilon})$ under plausible complexity theory assumptions is also known [38]. Thus, a generalization of VC on hypergraphs is well studied. However, there is very little study of FVS on hypergraphs. The only known algorithmic result is a factor d approximation for FVS on d-hypergraphs [55]. Upper bounds on minimum fvs in 3-uniform linear hypergraphs are studied in [39].

The objective of this chapter is to study the hypergraph variant of the FEEDBACK VERTEX SET problem from the view point of Parameterized Complexity.

One of the main reasons for the lack of study of FVS on hypergraphs is that it is not quite as natural to define the generalization of FVS in hypergraphs, as it is for the case of VC (generalizing to HS and d-HS) in hypergraphs. To generalize the notion of fvs to hypergraphs, we need to have notions of *cycles* and *forests* in hypergraphs. For cycles, we use the same notion as that in graph theory [39]: a cycle in a hypergraph H is a sequence $(v_0, e_0, v_1, \ldots, v_\ell, e_\ell, v_0)$ such that $v_0, \ldots, v_\ell$ are distinct vertices, $e_0, \ldots, e_\ell$ are distinct hyperedges, $\ell \geq 1$ and $v_i, v_{(i+1) \mod (\ell+1)} \in e_i$ for any $i \in \{0, \ldots, \ell\}$. Given the above definition of cycle, a subset S of vertices in a hypergraph H is called a *feedback vertex set*, if there does not exist a cycle in the hypergraph obtained after *deleting* vertices in S. The next natural question is what do we mean by *deletion* of a vertex in a hypergraph. There are two ways to define the vertex deletion operation in hypergraphs:

1. *Strong deletion* or simply *deletion* of a vertex v implies deleting v along with all the hyperedges containing the vertex v.
2. *Weak deletion* of a vertex v implies deleting v without deleting the hyperedges that contain v. That is, the hypergraph H' obtained after weak deletion of a vertex v from H has vertex set $V(H)$ and edge set $\{e \in E(H) \colon v \notin e\} \cup \{e \setminus \{v\} \colon e \in E(H), v \in e, |e| > 2\}$.

For a hypergraph H we use the notation $H - S$ to denote the graph obtained after (weak/strong) deletion of the vertices in S. Consequently, there are two ways one may define the FEEDBACK VERTEX SET problem – WEAK FVS and STRONG FVS.

Given a hypergraph H, the incidence graph G corresponding to H is the bipartite graph with bipartition $V(G) = A \uplus B$ where $A = V(H)$ and $B = E(H)$, and for any $v \in V(H)$ and $e \in E(H)$, ve is an edge in G if and only if $v \in e$ in H. Observe that WEAK FVS corresponds to finding a fvs S in G of size at most k, such that $S \subseteq A$ and $G - S$ is a forest.

Using the best known algorithm for WEIGHTED FVS [3] running in $3.618^k n^{\mathcal{O}(1)}$ time, we can solve WEAK FVS in $3.618^k n^{\mathcal{O}(1)}$ time, by transforming the problem to WEIGHTED FVS. To transform WEAK FVS to WEIGHTED FVS we assign every vertex in B a weight of $k+1$, every vertex in A a weight of 1. Now the problem of finding an fvs of weight at most k will be equivalent to solving WEAK FVS for the original hypergraph. Thus WEAK FVS is not challenging as a problem.

Hence, we only consider FVS on hypergraphs with respect to *strong deletion*. In particular, we study HYPERGRAPH FEEDBACK VERTEX SET (HFVS). Here, given an n-vertex hypergraph H and a positive integer k, the objective is to check whether there exists a set $S \subseteq V(H)$ of size at most k, such that $H - S$ is acyclic. As in the case of HS, it is expected that HFVS is W[2]-hard and this can be proven using a parameter preserving reduction from SET COVER (which is "equivalent" to HS). We prove the following theorem in Section 7.3.

Theorem 7.1.1. HFVS *is* W[2]-hard *when parameterized by k.*

Theorem 7.1.1 is not surprising as such a generalization of even VC is W[2]-hard.

FVS is a deeply studied problem in Parameterized Complexity, and thus, we tried to generalize the existing algorithms as much as possible. However, considering the problem on general hypergraphs is pushing it too far (Theorem 7.1.1). This motivated us to look for families of hypergraphs, which are a strict generalizations of graphs and where FVS turns out to be tractable. Specifically, we study the problem for the cases when the input is restricted to *linear hypergraphs* and *d-hypergraphs*.

A hypergraph H is linear if $|e \cap e'| \leq 1$ for any two distinct hyperedges $e, e' \in E(H)$. We show that for both these families, HFVS admits fixed parameter tractable (FPT) algorithms. Our main result is a randomized algorithm for the case when the input hypergraph is linear, and the size of the hyperedges is not bounded. Thus our positive results are the following.

Theorem 7.1.2. *There exists a deterministic algorithm for* HFVS *on d-hypergraphs,*

running in time $d^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$.

Theorem 7.1.3. *There exists an* $\mathcal{O}^\star(2^{\mathcal{O}(k^3 \log k)})$ *time*[1] *randomized algorithm for* HFVS *on linear hypergraphs, which produces a false negative output with probability at most* $\frac{1}{n^{\mathcal{O}(1)}}$, *and no false positive output.*

The restriction to linear hypergraphs corresponds to exclusion of C_4 or $K_{2,2}$ in the corresponding incidence graph. $K_{i,j}$ refers to the complete bipartite graph with partitions of sizes i and j. There has been extensive work on RED-BLUE DOMINATING SET for $K_{i,j}$ free graphs [30, 53, 101, 104]. Theorem 7.1.3 can be viewed as an analog of RED-BLUE DOMINATING SET results for $K_{2,2}$ free graphs.

The starting point of both the above mentioned algorithms (Theorems 7.1.2 and 7.1.3) is recasting HFVS as an appropriate problem on the incidence graph G of the given hypergraph H. Proof of Theorem 7.1.3 starts with the observation that for any subset $S \subseteq V(H)$, $H - S$ is acyclic if and only if $G - N_G[S]$ is acyclic. Consequently, HFVS is same as the following problem (see Lemma 41 in appendix for proof).

DOMINATING FVS ON BIPARTITE GRAPHS (DFVSB) **Parameter:** k
Input: A bipartite graph G with bipartition $V(G) = A \uplus B$ and $k \in \mathbb{N}$.
Question: Is there a subset $S \subseteq A$ of size at most k such that $G - N_G[S]$ is acyclic?

For a bipartite graph $G = (A \uplus B, E)$, we say that a subset $S \subseteq A$ is a *dominating feedback vertex set* for G if $G - N[S]$ is acyclic. Let G be the incidence graph of a hypergraph H. Then, notice that H is a d-hypergraph if and only if $\max_{e \in E(H)} d_G(e) \leq d$. Also, H is linear if and only if G is C_4-free. As a result HFVS on d-hypergraphs and linear hypergraphs are equivalent to DFVSB on bipartite graphs $G = (A \uplus B, E)$ with $\max_{w \in B} d(w) \leq d$ and on C_4-free bipartite graphs, respectively.

Theorem 7.1.2 shows that for d-hypergraphs, HFVS is similar to the case of d-HS. Proof of Theorem 7.1.2 utilizes iterative compression and the compression step involves

[1] Polynomial dependency on n is hidden in $\mathcal{O}^\star$ notation.

a branching strategy that uses a measure more generalized compared to the one used in known FVS algorithms for undirected graphs.

Our proof for Theorem 7.1.3 is inspired by the randomized algorithm of Becker et al. [8] that runs in $\mathcal{O}(4^k n^{\mathcal{O}(1)})$ time and the branching algorithm for POINT LINE COVER by Langerman and Morin [75]. The algorithm of Becker et al. [8] first preprocesses the input graph and transforms it into a graph with minimum degree at least 3 and then shows that for any fvs, at least half the edges in a preprocessed graph are incident to the vertex set of the fvs. This immediately implies the following algorithm: "*pick an edge uniformly at random, then pick a vertex that is an endpoint of this edge uniformly at random and add it to a solution, and recurse*". Let G be the incidence graph of a hypergraph H. First we preprocess G and show that in the preprocessed graph (say G) for any dominating feedback vertex set S of size at most k, at least $1/\mathsf{poly}(k)$ [2] fraction of all the edges are incident to $N[S]$. We call this property α-covering, with α being $\mathsf{poly}(k)$. Let S be a fixed fvs of size at most k. We now compute the probability of finding S. Note that if we randomly pick an edge f (that is, pick an edge from graph G uniformly at random and then select f as the hyperedge incident to the selected edge), then with probability $1/\mathsf{poly}(k)$ there exists a vertex incident to f that is contained in S. However, unlike the case of FVS in graphs, here we cannot randomly select a vertex from f, as the size of f could be independent of k. However, for now let us assume that we can preprocess $G-f$ such that α-covering property holds even after we delete f from G. We assume that α-covering property holds recursively after each iteration of preprocessing. Suppose we do this process k^2+1 times. Then we have a collection of hyperedges $\mathscr{F} = \{f_1, \ldots, f_{k^2+1}\}$ such that each of them has a non-trivial intersection with S. Observe that the pairwise intersection of these hyperedges cannot be more than one, since G excludes C_4 as a subgraph (H being a linear hypergraph). However, S is a solution of size at most k, and hence there exist $k+1$ hyperedges $f'_1, \ldots, f'_{k+1}$ among $\mathscr{F}$ such that $|f'_i \cap f'_j| = \{v\}$, $i \neq j$ for some $v \in A = V(H)$. This implies that v must belong to S, as each of $f'_1, \ldots, f'_{k+1}$ has a non-trivial intersection

[2] poly denotes a polynomial function.

with S and if we don't pick v, then every solution is of size at least $k+1$. Hence, we delete v along with all those edges in H that v participates in, and recursively find a solution of size $k-1$ in the reduced hypergraph.

However, unlike the case with FVS for graphs, in HFVS we cannot delete degree one vertices or contract degree 2 vertices directly. When we delete a hyperedge, we need to *remember* that we are seeking a solution that is a dominating feedback vertex set as well as a hitting set for the selected set. To implement this idea in our algorithm, we maintain a family $\mathscr{F}$ such that our solution is a dominating feedback vertex set for G as well as a hitting set for $\mathscr{F}$. We exploit the fact that $|\mathscr{F}| \leq k^2+1$ and design reduction rules that get rid of a certain degree one vertices and shorten degree 2 paths, as well as caterpillars (defined later) like degree 2 paths. We can show that after these reduction rules are performed, the α-covering property holds for the preprocessed graph, α being $\mathsf{poly}(k)$.

7.2 Preliminaries

For a positive integer $\ell \in \mathbb{N}$, we use $[\ell]$ to denote the set $\{1,2,\ldots,\ell\}$. We use the term graph to denote a simple graph without multiple edges, loops and labels. For the notations related to graphs that are not explicitly stated here, we refer to the book [40]. For a graph G and a subset of vertices $U \subseteq V(G)$, $N_G(U)$ and $N_G[U]$ denote the open neighborhood and closed neighborhood of U, respectively. That is, $N_G(U) = \{v \in V(G) \colon u \in U \text{ and } uv \in E(G)\} \setminus U$ and $N_G[U] = N_G(U) \cup U$. If $U = \{u\}$, then we write $N_G(u) = N_G(U)$ and $N_G[u] = N_G[U]$. Also, we omit the subscript G, if the graph in consideration is clear from the context. For a graph G, a vertex subset $X \subseteq V(G)$, and an edge subset $F \subseteq E(G)$, we use $G[X]$, $G-X$, and $G-F$ to denote the graph induced by X, the graph induced by $V(G) \setminus X$, and the graph with vertex set $V(G)$ and edge set $E(G) \setminus F$, respectively. Moreover, if $X = \{v\}$, then we write $G-v = G-X$. For a graph G, $X, Y \subseteq V(G)$, and $X \cap Y = \emptyset$, $E(X,Y) \subseteq E(G)$ denotes the set of edges in G whose one endpoint is in X and the other one is in Y. For a graph G

and a non-edge uv in G, we use $G+uv$ to denote the graph with vertex set $V(G)$ and edge set $E(G)\cup\{uv\}$. A path P in a graph G is a sequence of distinct vertices $u_1\dots u_\ell$ such that for all $i\in[\ell-1]$, $u_iu_{i+1}\in E(G)$. We say that a path $P=u_1\dots u_\ell$ in a graph G is a *degree two path* in G, if for each $i\in[\ell]$, the degree of u_i in G, denoted by $d_G(u_i)$, is equal to 2. For a path/cycle P, we use $V(P)$ to denote the set of vertices present in P. A triangle is a cycle consisting of exactly 3 edges. A bipartite graph $G=(A\uplus B,E)$ is called a d-bipartite graph if $d_G(b)\leq d$ for all $b\in B$. For two hypergraphs H_1 and H_2, $H_1\cup H_2$ denotes the hypergraph with the vertex set $V(H_1)\cup V(H_2)$ and the edge set $E(H_1)\cup E(H_2)$.

7.3 Feedback Vertex Sets on General Hypergraphs

In order to prove Theorem 7.1.1 we give a polynomial time parameter preserving reduction from SET COVER to HFVS. In SET COVER (SC), we are given a universe U, a family $\mathscr{F}$ of sets over U, and a positive integer k, and the question is whether there is a subfamily $\mathscr{F}'\subseteq\mathscr{F}$ of size at most k, such that $\bigcup_{F\in\mathscr{F}'}F=U$. It is well known that SET COVER is W[2]-hard [29, Theorem 13.28].

Given an instance $(U,\mathscr{F},k)$ of SC, we construct an instance (H,k) of HFVS as follows. For each element $u\in U$, let X_u be the family of sets in $\mathscr{F}$ that contain u. For each $F\in\mathscr{F}$, we add a vertex w_F in H. Furthermore, for each $u\in U$, we add $2(k+1)$ vertices $\{u_1,u'_1,\dots,u_{k+1},u'_{k+1}\}$ in H. Hence, $V(H)=\{w_F\mid F\in\mathscr{F}\}\cup\{u_1,u'_1,\dots,u_{k+1},u'_{k+1}\mid u\in U\}$. Now, we explain the construction of hyperedges of H. For each $u\in U$, we introduce a hyperedge $e_u=\{w_F\mid F\in X_u\}$ containing vertices corresponding to the sets in X_u. Also, for each $u\in U$, we add hyperedges $e_u\cup\{u_i\}$, $\{u_i,u'_i\}$, $e_u\cup\{u'_i\}$, for all $i\in[k+1]$. This completes the construction. Towards the proof of Theorem 7.1.1, we give the following lemma.

Lemma 40. *$(U,\mathscr{F},k)$ is a* yes*-instance of* SC *if and only if (H,k) is a* yes*-instance of* HFVS.

Proof. In the forward direction, let $\mathscr{S}$ be a solution to $(U,\mathscr{F},k)$ of SC. We claim that $Z = \{w_F \mid F \in \mathscr{S}\}$ is a feedback vertex set of size at most k in H. Since $|\mathscr{S}| \leq k$, we have that $|Z| \leq k$. Next, we prove that Z is a feedback vertex set in H. Since $\mathscr{S}$ is a set cover, the only hyperedges of H present in $H - Z$ are $\{\{u_i, u'_i\} \mid u \in U, i \in [k+1]\}$. Notice that $\{\{u_i, u'_i\} \mid u \in U, i \in [k+1]\}$ are pairwise disjoint. This implies that $H - Z$ is acyclic.

In the reverse direction, let Z be a solution to (H,k) of HFVS. Let $Z' = Z \setminus \{u_1, u'_1, \ldots, u_{k+1}, u'_{k+1} \mid u \in U\}$. That is, Z' contains only those vertices of Z that correspond to some set in $\mathscr{F}$. Let $\mathscr{S} = \{F \mid w_F \in Z'\}$. Since $|Z'| \leq |Z| \leq k$, we have that $|\mathscr{S}| \leq k$. Next we claim that $\mathscr{S}$ is a set cover of $(U,\mathscr{F},k)$. Towards that, we choose an arbitrary element $u \in U$ and prove that there is a set $F \in \mathscr{S}$ which contains u. Let J be an arbitrary set in $\mathscr{F}$ such that $u \in J$. Notice that there are $k+1$ triangles $(w_J, e_u \cup \{u_i\}, u_i, \{u_i, u'_i\}, u'_i, e_u \cup \{u'_i\}, w_J), 1 \leq i \leq k+1$, in H. This implies that at least one vertex w_F in e_u must belong to the feedback vertex set Z (and Z'). However, u belongs to F and F is in $\mathscr{S}$. Hence u is covered by $\mathscr{S}$. This completes the proof. □

7.4 Equivalence between HFVS and DFVSB

Lemma 41. *(H,k) is a* yes*-instance of* HFVS *if and only if $(G = (A \uplus B, E'),k)$ is a* yes*-instance of* DFVSB*, where G is the incidence graph of the hypergraph H.*

Proof. In forward direction, let S be a solution to (H,k) of HFVS. We claim that S is also a solution to $(G = (A \uplus B, E'),k)$ of DFVSB. Suppose not. Then, there exists a cycle $C = v_1 e_1 \ldots v_\ell e_\ell v_1$ in the graph $G - N_G[S]$. This implies that $e_1, \ldots, e_\ell$ are hyperedges in $H - S$, and $\{v_1, \ldots, v_\ell\} \subseteq V(H) \setminus S$. Then $(v_1, e_1, \ldots, v_\ell, e_\ell, v_1)$ is a cycle in the hypergraph $H - S$. This is a contradiction to the assumption that S is a solution to (H,k).

In reverse direction, let S' be a solution to (G,k) of DFVSB. We claim that S' is also a solution to (H,k) of HFVS. Suppose not. Then, there exists a cycle $C =$

$(v_1, e_1, \ldots, v_\ell, e_\ell, v_1)$ in the hypergraph $H - S'$. This implies that $\{v_1, \ldots, v_\ell\} \subseteq A \setminus S'$ and $\{e_1, \ldots, e_\ell\} \subseteq B \setminus N_G(S')$. Therefore, $v_1 e_1 \ldots v_\ell e_\ell v_1$ is a cycle in $G - N_G[S']$, which is a contradiction to the assumption that S' is a solution to (G,k). □

7.5 Feedback Vertex Sets on d-Hypergraphs: Proof of Theorem 7.1.2

In this section we design an FPT algorithm for HFVS on d-hypergraphs. Towards this, we will prove the following result about DFVSB, from which Theorem 7.1.2 will follow as a corollary.

Theorem 7.5.1. *There is a deterministic algorithm for* DFVSB *running in time* $\mathcal{O}(2^{7k} d^{2k+1} n(n+m) + n^2(n+m))$*, where the input is a bipartite graph* G *with bipartition* $V(G) = A \uplus B$*, and* $d = \max_{b \in B} d_G(b)$.

Towards designing an FPT algorithm for DFVSB, we use the well-known iterative compression technique [29, Chapter 4]. Usually, the primary step in the technique of iterative compression involves solving a "disjoint compression version" of the problem. In our case, the disjoint compression version of the problem is defined as follows.

d-DISJOINT DOMINATING BOUNDED BIPARTITE FVS (d-DDBB-FVS)
Input: A d-bipartite graph $G = (A \uplus B, E)$, a positive integer k, and a vertex subset $W \subseteq A$ such that $G - N[W]$ is acyclic.
Question: Is there a set $S \subseteq A \setminus W$ of at most k vertices such that $G - N[S]$ is acyclic?

We denote an instance of d-DDBB-FVS as (G,k,W), where G is the input graph with bipartition $A \uplus B$, k is the parameter (the solution size), and W is a set such that for a solution S, it holds that $S \subseteq A \setminus W$. The main result of the section is the following lemma.

Lemma 42. *Given an instance* $((A \uplus B, E), k, W)$ *of* d-DDBB-FVS*, there exists an algorithm that gives a solution in time* $\mathcal{O}((8d)^{k+\gamma(G_W)}(n+m) + n(n+m))$*, where* $d =$

$\max_{w \in B} d(w)$, $n = |V(G)|$, $m = |E(G)|$, *and* $\gamma(G_W)$ *is the number of connected components in the subgraph* $G_W = G[W \cup \{b \in B \colon N(b) \subseteq W\}]$.

Assuming Lemma 42 one can prove Theorem 7.5.1, somewhat similar to the way it is done for FVS on graphs (see Section 4.1 in [29]). We use the following observation in the proof of Theorem 7.5.1.

Observation 7.5.2. *Let* $(G = (A \uplus B, E), k)$ *be an instance of* DFVSB*, and* $B' \subseteq B$*. If* $(G' = (A \uplus B', E(A, B')), k)$ *is a* no-*instance of* DFVSB*, then* $(G = (A \uplus B, E), k)$ *is a* no-*instance of* DFVSB.

Proof. Any solution to $(G = (A \uplus B, E), k)$ is also a solution to $((A \uplus B', E(A, B')), k)$. □

Now we give a proof sketch of Theorem 7.5.1 assuming Lemma 42.

Proof sketch of Theorem 7.5.1. We employ the method of iterative compression to prove Theorem 7.5.1. Towards that, we iteratively apply Lemma 42. Let $(G = (A \uplus B, E), k)$ be the input of DFVSB. Let $B = \{b_1, \ldots, b_r\}$. If $r \leq k+1$, then any subset $A' \subseteq A$ of size at most $r-1$ that contains a neighbor of b_i for all $i \in [r-1]$ is a solution to (G, k). That is, if $r \leq k+1$, then (G, k) is a yes-instance. Otherwise, we proceed as follows.

Initially we consider the instance $J_1 = (G_1 = (A \uplus B_1), k)$ of DFVSB, where $B_1 = \{b_1, \ldots, b_{k+2}\}$. Let $W_1 = \{v_1, \ldots, v_{k+1}\}$ be an arbitrary subset of A such that $N(b_j) \cap W_1 \neq \emptyset$ for all $j \in [k+1]$. Clearly, W_1 is a dominating feedback vertex set of size $k+1$ for G_1. To compute a dominating feedback vertex set of size at most k, for each subset $S \subseteq W_1$ of size at most k (a potential guess of the intersection of a hypothetical solution with W_1), we use Lemma 42 to check whether there exists a solution to the instance $(G_1' = G_1 - N[S], k - |S|, W_1 \setminus S)$ of d-DDBB-FVS. If no such solution exists for any choice of the subset of W_1, then clearly J_1 is a no-instance of DFVSB due to observation 7.5.2.

Otherwise, if there is a subset $S_1 \subseteq W_1$ of size at most k, such that Q_1 is a solution for $(((A \setminus S_1) \uplus (B_1 \setminus N(S_1)), E(A \setminus S_1, B_1 \setminus N(S_1))), k - |S_1|, W_1 \setminus S_1)$ of d-DDBB-FVS,

then $S_1 \cup Q_1$ is a solution of size at most k for the instance J_1. Next, we construct an instance $J_2 = (G_2 = (A \uplus B_2, E(A,B_2)),k)$ of DFVSB, where $B_2 = \{b_1,\ldots,b_{k+3}\}$. Let $W_2 = S_1 \cup Q_1 \cup \{v\}$, where v is an arbitrary vertex in $N(b_{k+3})$. Notice that $G_2 - N[W_2]$ is a subgraph of $G_1 - N[S_1 \cup Q_1]$ which is a forest. That is, W_2 is a dominating feedback vertex set of G_2 of size at most $k+1$. Now we repeat the same process as described above to "compress" the solution size of J_2 to at most k. At each iteration, if there exists a solution W_i of size at most k for the instance J_i, then in step $i+1$, $W_i \cup \{v\}$ is a dominating feedback vertex set for $G_{i+1} = (A \uplus B_{i+1}, E(A,B_{i+1}))$, where $B_{i+1} = B_i \cup \{b_{k+2+i}\}$ and $v \in N(b_{k+2+i})$, and we continue the same process.

Finally, notice that $J_{r-(k+1)}$ is actually the input instance (G,k), and we get a solution to $J_{r-(k+1)}$ at the end of the algorithm (if (G,k) is a yes-instance). More formally, at step $i \in [r-(k+1)]$, we have an instance $J_i = (G_i = (A \uplus B_i, E(A,B_i)),k)$, where $B_i = \{b_1,\ldots,b_{k+1+i}\}$, and a dominating feedback vertex set W_i' of G_i of size at most $k+1$. Then, by applying Lemma 42 at most 2^{k+1} times we obtain a solution W_i of size at most k for the instance J_i (if it exists). If there does not exist a solution for J_i, then (G,k) is a no-instance.

Since we apply Lemma 42 at most $2^{k+1}|B| - (k+1)$ times and the number of connected components of G_W in each application of Lemma 42 is at most $k+1$, the total running time is upper bounded by $\mathcal{O}(2^k(8d)^{2k+1}n(n+m) + n^2(n+m)) = \mathcal{O}(2^{7k}d^{2k+1}n(n+m) + n^2(n+m))$, where $n = |V(G)|$ and $m = |E(G)|$. □

The rest of the section is devoted to the proof of Lemma 42. Towards proving Lemma 42, we design a branching algorithm consisting of three branching rules and some simple reduction rules. To bound the running time, we define a *measure* associated with an instance of d-DDBB-FVS, and this measure decreases by at least one during each application of the branching rules. It does not increase during the application of any of the reduction rules. Moreover, the number of children for each node in the branching tree is bounded by $\mathcal{O}(d)$. For an instance $(G = (A \uplus B, E), k, W)$ of d-DDBB-FVS, recall that

$G_W = G[W \cup \{b \in B\colon N_G(b) \subseteq W\}]$, and $\gamma(G_W)$ is the number of connected components in G_W. We define the measure associated with the instance (G,k,W) of d-DDBB-FVS as,

$$\mu(G,k,W) = k + \gamma(G_W)$$

For a reduction rule that takes an instance (G,k,W) of d-DDBB-FVS and outputs another instance (G',k',W') of d-DDBB-FVS, we say that the reduction rule is safe if the following holds: (i) (G,k,W) is a yes-instance if and only if (G',k',W') is a yes-instance, and (ii) $\mu(G',k',W') \leq \mu(G,k,W)$. A branching rule for d-DDBB-FVS, takes an instance (G,k,W) and outputs a collection of instances $(G_1,k_1,W_1),\ldots,(G_\ell,k_\ell,W_\ell)$. We say that the branching rule is safe if the following holds: (i) (G,k,W) is a yes-instance if and only if (G_i,k_i,W_i) is a yes-instance for some $i \in [\ell]$, and (ii) for each $i \in [\ell]$, $\mu(G_i,k_i,W_i) < \mu(G,k,W)$.

Reduction Rule 21. *Let (G,k,W) be an instance of* d-DDBB-FVS. *If $k = 0$ and G is not acyclic, then return that (G,k,W) is a* no-*instance of* d-DDBB-FVS.

Reduction Rule 22. *Let (G,k,W) be an instance of* d-DDBB-FVS. *If G is acyclic and $k \geq 0$, then return $\emptyset$ and STOP.*

The correctness of the above reduction rules follows from the fact that (G,k,W) is a yes-instance of d-DDBB-FVS and $\emptyset$ is a solution to (G,k,W).

Reduction Rule 23. *Let (G,k,W) be an instance of* d-DDBB-FVS. *Let $v \in V(G)$ be a vertex of degree* 0 *in G. Then, output $(G-v,k,W \setminus \{v\})$.*

It is easy to see that the above reduction rules are safe and can be applied in polynomial time.

Reduction Rule 24. *Let $(G = (A \uplus B, E),k,W)$ be an instance of* d-DDBB-FVS *and $b \in B$ be a vertex of degree* 1 *in G. Then, output $(G-b,k,W)$.*

Lemma 43. *Reduction Rule 24 is safe.*

Proof. Since $d_G(b) = 1$, there is no cycle in G containing b. Therefore, any solution to $(G-b,k,W)$ is also a solution to (G,k,W) and vice versa. Let $G' = G-b$. Since $d_G(b) \leq 1$, $\gamma(G'_W) \leq \gamma(G_W)$. Therefore, $\mu(G',k,W) \leq \mu(G,k,W)$ and Reduction Rule 24 is safe. □

Reduction Rule 25. *Let* $(G = (A \uplus B, E), k, W)$ *be an instance of* d-DDBB-FVS *and* $v \in A \setminus W$ *be a vertex of degree* 1 *in* G. *Let* $N_G(v) = \{b\}$. *Moreover, either* $N_G(b) \setminus (W \cup \{v\}) \neq \emptyset$ *or* $d_G(b) = 2$. *Then, output* $(G-v,k,W)$.

Lemma 44. *Reduction Rule 25 is safe.*

Proof. First consider the case $N_G(b) \setminus (W \cup \{v\}) \neq \emptyset$. Since $d_G(v) = 1$, any solution to $(G-v,k,W)$ is also a solution to (G,k,W). Now suppose that, (G,k,W) is a yes-instance. Let u be an arbitrary vertex in $N_G(b) \setminus (W \cup \{v\})$ and $G' = G-v$. First we claim that there is a solution S to (G,k,W) that does not contain v. If there exists a solution S' to (G,k,W) that contains v, then $S^\star = (S' \setminus \{v\}) \cup \{u\}$ is a solution to (G,k,W), because $N_G(v) \subseteq N_G(u)$ and $d_G(v) = 1$. Let S be a solution to (G,k,W) such that $v \notin S$. Then, S is also a solution to $(G' = G-v,k,W)$ because $G' - N_{G'}[S] = (G - N_G[S]) - v$, and $(G - N_G[S])$ is acyclic. Notice that $G'_W = G_W$. Therefore, $\mu(G',k,W) \leq \mu(G,k,W)$.

Next, we consider the case $d_G(b) = 2$. Here, there is no cycle in G that contains either b or v. This implies that, if S is a solution to (G,k,W), then $S \setminus \{v\}$ is a solution to $(G-v,k,W)$. Since $d_G(v) = 1$, any solution to $(G' = G-v,k,W)$ is also a solution to (G,k,W). Also, since $G'_W = G_W$, we have that $\mu(G',k,W) \leq \mu(G,k,W)$. □

Reduction Rule 26. *Let* (G,k,W) *be an instance of* d-DDBB-FVS. *Let* $b_1v_1b_2v_2b_3v_3b_4$ *be a path in* G *such that* $v_1b_2v_2b_3v_3$ *is a degree two path in* G, $\{b_1,\ldots,b_4\} \subseteq B$ *and* $\{v_1,v_2,v_3\} \subseteq A \setminus W$. *Now, let* G' *be the graph obtained by deleting the vertices* b_2,v_2 *from* G *and adding a new edge* v_1b_3, *that is,* $G' = (G - \{v_2,b_2\}) + v_1b_3$. *Then, output* (G',k,W).

Lemma 45. *Reduction Rule 26 is safe.*

Proof. First, we prove that (G,k,W) is a yes-instance of d-DDBB-FVS if and only if (G',k,W) is a yes-instance of d-DDBB-FVS. In the forward direction, let S be a solution to (G,k,W) of d-DDBB-FVS. Suppose that, $v_2 \notin S$. Then, we claim that S is also a solution of (G',k,W). Suppose not, then there exists a cycle C in $G' - N_{G'}[S]$. If C does not contain the edge v_1b_3, then C is also a cycle in $G - N_G[S]$, which is a contradiction. Therefore, C contains the edge v_1b_3. But, then we get a cycle in $G - N_G[S]$ by replacing the edge v_1b_3 in C by the path $v_1b_2v_2b_3$. This is a contradiction to the assumption that S is a solution to (G,W,k). Now, consider the case $v_2 \in S$. Then, $S' = (S \setminus \{v_2\}) \cup \{v_1\}$ is a solution to (G',k,W) because $S' \cap W = \emptyset$ and any cycle in G which contains any of the vertices in $\{b_2,v_2,b_3\}$ also contains v_1.

For the backward direction, let $S^\star$ be a solution to (G',k,W) of d-DDBB-FVS. Clearly, $S^\star \subseteq A \setminus W$. We claim that $S^\star$ is also a solution to (G,k,W). Suppose not. Then, there exists a cycle C in $G - N_G[S^\star]$. If C does not contain any edges from $\{v_1b_2, b_2v_2, v_2b_3\}$, then C is also a cycle in $G' - N_{G'}[S^\star]$, which is a contradiction. Therefore, at least one edge from $\{v_1b_2, b_2v_2, v_2b_3\}$ is part of C. Then, since $v_1b_2v_2b_3v_3$ is a degree two path in G, $b_1v_1b_2v_2b_3v_3b_4$ is a subpath in C. Then, we get a cycle C' in $G' - N_{G'}[S^\star]$ by replacing the subpath $v_1b_2v_2b_3$ in C by v_1b_3. This is a contradiction to the assumption that $S^\star$ is a solution to (G',k,W). Hence, $S^\star$ is also a solution to (G,k,W).

Next, we prove that $\mu(G',k,W') \leq \mu(G,k,W)$. Since $v_1,v_2,v_3 \notin W$, we have that $b_1,b_2,b_3,b_4 \notin V(G_W)$. Therefore, we have that $G_W = G'_{W'}$ and hence, $\mu(G',k,W') = \mu(G,k,W)$. This completes the proof of the lemma. □

Branching Rule 1. *Let (G,k,W) be an instance of d-DDBB-FVS and let $b \in B$ be a vertex such that $N_G(b) \setminus W \neq \emptyset$ and $|N_G(b) \cap W| \geq 2$. Let $z,z' \in N_G(b) \cap W$ be two distinct vertices and $N_G(b) \setminus W = \{u_1,\ldots,u_\ell\}$. If z and z' are in the same connected component of G_W, then we branch into the following instances: $(G - N[u_1],k-1,W),\ldots,(G - N[u_\ell],k-1,W)$. If z and z' are in two distinct connected components of G_W, then we branch into the following*

instances: $(G-N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$*, and* $(G,k,W\cup\{u_1,\ldots,u_\ell\})$.

Lemma 46. *Branching Rule 1 is safe.*

Proof. First consider the case that z and z' are in the same connected component of G_W. If (G,k,W) is a no-instance, then clearly all the instances $(G-N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$ are no-instances. Since z and z' are in the same connected component of G_W, there is a cycle C in $G[V(G_W)\cup\{b\}]$. Also, notice that $N_G(V(C)\cap B)\setminus W\subseteq\{u_1,\ldots,u_\ell\}$. That is, if (G,k,W) is a yes-instance, then any solution will contain a vertex from $\{u_1,\ldots,u_\ell\}$. Therefore, if (G,k,W) is a yes-instance, then at least one of the instances $(G-N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$ is a yes-instance. Now we prove that $\mu(G-N[u_i],k-1,W)\leq\mu(G,k,W)-1$, for all $i\in[\ell]$. Towards that, we fix an arbitrary $i\in[\ell]$. Let $G'=G-N[u_i]$. Since $u_i\in A\setminus W$, $G_W=G'_W$. This implies that, $\gamma(G'_W)=\gamma(G_W)$. Therefore, $\mu(G',k-1,W)=k-1+\gamma(G'_W)=k+\gamma(G_W)-1=\mu(G,k,W)-1$.

Next, consider the case that z and z' are in two different connected components of G_W. If (G,k,W) is a no-instance, then clearly all the instances $(G-N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$, and $(G,k,W\cup\{u_1,\ldots,u_\ell\})$ are no-instances. Suppose that, (G,k,W) is yes-instance. Let S be a solution to (G,k,W). If $S\cap\{u_1,\ldots,u_\ell\}\neq\emptyset$, then at least one of $(G-N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$ is a yes-instance. Otherwise, S is a solution to $(G,k,W\cup\{u_1,\ldots,u_\ell\})$. The proof of $\mu(G-N[u_i],k-1,W)\leq\mu(G,k,W)-1$ for all $i\in[\ell]$, given in the above paragraph holds in this case as well. Finally, we prove that $\mu(G,k,W\cup\{u_1,\ldots,u_\ell\})\leq\mu(G,k,W)-1$. Note that, it is enough to prove that $\gamma(G_{W'})\leq\gamma(G_W)-1$, where $W'=W\cup\{u_1,\ldots,u_\ell\}$. Observe that, each connected component in $G_{W'}$ contains a vertex from W', as Reduction Rule 23 is no longer applicable. Moreover, G_W is a subgraph of $G_{W'}$ and there is a connected component in $G_{W'}$ containing z and z', because $z,z'\in N_G(b)$ and $b\in V(G_{W'})$. Also, notice that in this case z and z' belong to different connected components in G_W. This implies that, $\gamma(G_{W'})\leq\gamma(G_W)-1$. This completes the proof of the lemma. □

Branching Rule 2. *Let* (G,k,W) *be an instance of* d-DDBB-FVS. *If there exists a*

path/cycle $P = b_0v_0 \ldots b_rv_rb_{r+1}$ *in* G*, such that* $\{v_0, \ldots, v_r\} \subseteq A \setminus W$*,* $0 \leq r \leq 6$*, and there is a cycle in the graph* $G[V(G_W) \cup V(P)]$*, then we branch into the following instances:* $(G - N[u_1], k-1, W), \ldots, (G - N[u_\ell], k-1, W)$*, where* $\{u_1, \ldots, u_\ell\} = N_G(\{b_0, \ldots, b_{r+1}\}) \setminus W$*.*

Lemma 47. *Branching Rule 2 is safe.*

Proof. If (G, k, W) is a no-instance, then clearly all the instances $(G - N[u_1], k - 1, W), \ldots, (G - N[u_\ell], k-1, W)$ are no-instances. Now, we prove that if (G, k, W) is a yes-instance, then at least one of the instances $(G - N[u_1], k-1, W), \ldots, (G - N[u_\ell], k-1, W)$ is a yes-instance. Notice that there exists a cycle C in $G[V(G_W) \cup V(P)]$. Therefore, any solution to (G, k, W) contains a vertex from $N_G(V(C) \cap B) \setminus W$. Since $N_G(b) \subseteq W$ for all $b \in B \cap V(G_W)$, we have that $N_G(V(C) \cap B) \setminus W \subseteq N(\{b_0, \ldots, b_{r+1}\}) \setminus W = \{u_1, \ldots, u_\ell\}$. Therefore, if (G, k, W) is a yes-instance, then at least one of the instances $(G - N[u_1], k-1, W), \ldots, (G - N[u_\ell], k-1, W)$ is a yes-instance as well.

Next, we prove that $\mu(G - N[u_i], k-1, W) = \mu(G, k, W) - 1$ for all $i \in [\ell]$. Towards that, we fix an arbitrary $i \in [\ell]$. Let $G' = G - N[u_i]$. Since $u_i \in A \setminus W$, we have that $G_W = G'_W$. Therefore, $\mu(G', k-1, W) = k - 1 + \gamma(G'_W) = k + \gamma(G_W) - 1 = \mu(G, k, W) - 1$. This completes the proof of the lemma. □

Branching Rule 3. *Let* (G, k, W) *be an instance of* d-DDBB-FVS*. Let* $P = b_0v_0, \ldots, b_rv_rb_{r+1}$ *be a path in* G*, such that* $0 \leq r \leq 6$ *and* $\{v_0, \ldots, v_r\} \subseteq A \setminus W$*. Let* z *and* z' *be two vertices in two distinct connected components of* G_W*. If there is path from* z *to* z' *in the graph* $G[V(G_W) \cup V(P)]$*, then we branch into the following instances:* $(G - N[u_1], k - 1, W), \ldots, (G - N[u_\ell], k-1, W)$*, and* $(G, k, W \cup \{u_1, \ldots, u_\ell\})$*, where* $\{u_1, \ldots, u_\ell\} = N_G(\{b_0, \ldots, b_{r+1}\}) \setminus W$*.*

Lemma 48. *Branching Rule 3 is safe.*

Proof. If (G, k, W) is a no-instance, then clearly all the instances $(G - N[u_1], k - 1, W), \ldots, (G - N[u_\ell], k-1, W)$ and $(G, k, W \cup \{u_1, \ldots, u_\ell\})$ are no-instances. Now we

prove that if (G,k,W) is a yes-instance, then at least one of the instances $(G - N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$ and $(G,k,W\cup\{u_1,\ldots,u_\ell\})$ is a yes-instance. Let S be a solution to (G,k,W). If $S\cap\{u_1,\ldots,u_\ell\}\neq\emptyset$, then at least one of $(G - N[u_1],k-1,W),\ldots,(G-N[u_\ell],k-1,W)$ is a yes-instance. Otherwise S is a solution to $(G,k,W\cup\{u_1,\ldots,u_\ell\})$.

Next, we prove that $\mu(G-N[u_i],k-1,W)\leq\mu(G,k,W)-1$, for all $i\in[\ell]$. Here, the proof follows the arguments similar to those in the proof of Lemma 47. Now we prove that $\mu(G,k,W\cup\{u_1,\ldots,u_\ell\})\leq\mu(G,k,W)-1$. Towards that, it is enough to prove that $\gamma(G_{W'})\leq\gamma(G_W)-1$, where $W'=W\cup\{u_1,\ldots,u_\ell\}$. Notice that each connected component in $G_{W'}$ contains a vertex from W'. Moreover, G_W is a subgraph of $G_{W'}$ and there is a connected component in $G_{W'}$ containing z and z', because $V(P)\subseteq V(G_{W'})$. Also, notice that by our assumption z and z' belong to different connected components in G_W. This implies that, $\gamma(G_{W'})\leq\gamma(G_W)-1$. This completes the proof of the lemma. □

Now we are ready to complete the proof of Lemma 42.

Proof of Lemma 42. We design a branching algorithm for the problem. Let (G,k,W) be an instance of d-DDBB-FVS. We prove that we can always apply either one of the reduction rules or one of the branching rules until we reach a solution or a no-instance. First we test if any of the Reduction Rules 21, 22, 23, 24, and 25 is applicable. This can easily be tested in linear time. If any of these reduction rules are applicable, we apply them. Next, we test whether Reduction Rule 26 is applicable. Towards that, let H be a graph obtained from G by deleting all the vertices in W and the vertices of degree at least 3 in G. Then, for any maximal path P such that the internal vertices of P have degree exactly two in G and $V(P)\cap W=\emptyset$, there exists a component in H which is an induced path containing all the vertices of P. Thus, we can identify such a path $P=b_1v_1b_2v_2b_3v_3b_4$ in G such that the internal vertices of P are degree exactly two in G and $V(P)\cap W=\emptyset$ (if it exists) in linear time. If such a path exists, then we apply Reduction Rule 26. Next, if Branching Rule 1 is

applicable, then we apply it. This can be done in linear time as well.

For rest of the proof, we assume that Reduction Rules 21–26, and Branching Rule 1 are not applicable on (G,k,W). We know that $F = G - N_G[W]$ is acyclic. Since $d_G(b) \geq 2$ for all $b \in B$ (because Reduction Rules 23 and 24 are not applicable) and $F = G - N_G[W]$, (i) any vertex $u \in V(F)$ with degree at most 1 in F (that is, $d_F(u) \leq 1$) belongs to $A \setminus W$. Now we claim that (ii) there is no vertex of degree zero in F. Suppose not. Let $v \in V(F)$ be such that $d_F(v) = 0$. Because of statement (i), we have that $v \in A \setminus W$. Since Reduction Rule 23 is not applicable, we have that $d_G(v) \geq 1$. If $d_G(v) = 1$, then $N_G(b) \setminus (W \cup \{v\}) = \emptyset$ and $d_G(b) > 2$, where $\{b\} = N_G(v)$, as Reduction Rules 24 and 25 are not applicable. This implies that, $N_G(b) \setminus W \neq \emptyset$ and $|N_G(b) \cap W| \geq 2$. As a result Branching Rule 1 is applicable, which is a contradiction. Thus, we have proven statement (ii).

Next we prove that (iii) for each $v \in V(F)$ such that degree of v is 1 in F, there is a vertex $b \in N_G(W)$ such that $vb \in E(G)$. Towards that, it is enough to prove that for each $v \in V(F)$ of degree 1 in F, $d_G(v) \geq 2$. If $d_G(v) = 1$, then $N_G(b) \setminus (W \cup \{v\}) = \emptyset$ and $d_G(b) > 2$, where $\{b\} = N_G(v)$, as Reduction Rules 24 and 25 are not applicable. This implies that, $N_G(b) \setminus W \neq \emptyset$ and $|N_G(b) \cap W| \geq 2$. As a result Branching Rule 1 is applicable, which is a contradiction. Thus, we have proven statement (iii).

Let Q be a path in F (of length more than 0) such that the end-vertices of Q have degree 1 in F, and all but at most one internal vertex of Q has degree exactly 2 in F. Any forest F containing at least one edge contains such a path and it can be computed in linear time. Since the end-vertices of Q have degree 1 in F, by statement (i), the end-vertices of Q belong to $A \setminus W$. Let $Q = v_0 b_1 \ldots b_\ell v_\ell$ for some $\ell \in \mathbb{N}$, where $\{v_1, \ldots, v_\ell\} \subseteq A \setminus W$ and $\{b_1, \ldots, b_\ell\} \subseteq B \setminus N_G(W)$. Due to statement (iii), there exist vertices $b, b' \in N_G(W)$ (not necessarily distinct), such that $bv_0, b'v_\ell \in E(G)$.

Case 1: $\ell \leq 6$**.** Let P be the path/cycle $bv_0 b_1 \ldots b_\ell v_\ell b'$. Note that, P is a cycle if $b = b'$ and P is a path if $b \neq b'$. If P is a cycle, then Branching Rule 2 is applicable and we apply it. Suppose that, $b' \neq b$. Notice that $b, b' \in N_G(W)$. This implies that, there exist vertices z

and z' in W, such that $bz, b'z' \in E(G)$. If z and z' belong to the same connected component in G_W, then either Branching Rule 2 is applicable, or Branching Rule 3 will be applicable due to existence of path P. We apply the branching rule accordingly.

Case 2: $\ell \geq 7$**.** Recall that, all but at most one vertex in $Q = v_0 b_1 \ldots b_\ell v_\ell$ has degree at most 2 in F. If all the vertices in Q have degree at most two in F, then either no vertex v_i, $i \in \{1,\ldots,3\}$ has a neighbor in $N(W)$ and Reduction Rule 26 is applicable, or there exists a vertex v_i, $i \in \{1,\ldots,3\}$, such that v_i has a neighbor in $N(W)$ and either Branching Rule 2, or Branching Rule 3 is applicable. Next, consider that there exists a vertex in Q with degree more than 2 in F. (a) A vertex in $\{v_1, v_2, v_3, b_1, b_2, b_3\}$ has degree more than 2 in F. (b) A vertex in $\{v_4, v_5, v_6, b_4, b_5, b_6, b_7\}$ has degree more than 2 in F. Without loss of generality let us assume (b) (Other case can be argued similarly). That is, each vertex in $\{v_1, v_2, v_3, b_1, b_2, b_3, \}$ has degree at most 2 in F. First, we prove that there exists $i \in \{1,\ldots,3\}$ such that $N_G(v_i) \cap N_G(W) \neq \emptyset$. Otherwise $v_1 b_2 v_2 b_3 v_3$ is a degree two path in G, and hence, Reduction Rule 26 is applicable, a contradiction to the assumption that none of the reduction rules are applicable.

Now, we fix $i \in \{1,\ldots,3\}$ such that $N_G(v_i) \cap N_G(W) \neq \emptyset$. Let $b^\star \in N_G(W)$ be such that $v_i b^\star \in E(G)$. Let $Q^\star$ be the subpath of Q between v_0 and v_i and $P^\star$ be the path $bQ^\star b^\star$. Clearly, due to existence of path $P^\star$, either Branching Rule 2 or Branching Rule 3 is applicable. We apply the branching rule accordingly.

Now we do the running time analysis. Let $n = |V(G)|$ and $m = |E(G)|$. Each application of a reduction rule takes linear time. Moreover, after each application of a reduction rule, the number of vertices in the graph drops by at least one. Therefore, the total time taken to apply all the reduction rules together in one branch of the branching tree is upper bounded by $\mathcal{O}(n(n+m))$. Each application of a branching rule takes linear time. The number of branches created during an application of Branching Rules 2 or 3 is at most $8d$. Moreover, after each application of Branching Rules 2 and 3, the measure associated with the instance drops by at least one. Therefore, the total running time is upper bounded by

$\mathcal{O}((8d)^{k+\gamma(G_W)}(n+m)+n(n+m))$. This concludes the proof. □

7.6 Feedback Vertex Sets on Linear Hypergraphs

In this section we design an FPT algorithm for HFVS on linear hypergraphs. Towards this, we prove the following result about DFVSB, from which Theorem 7.1.3 follows as a corollary.

Theorem 7.6.1. *There exists an $\mathcal{O}^\star(2^{\mathcal{O}(k^3 \log k)})$ time randomized algorithm for* DFVSB *on C_4-free bipartite graphs, which produces a false negative output with probability at most $\frac{1}{n^{\mathcal{O}(1)}}$, and no false positive output.*

To prove Theorem 7.6.1, we need to take a detour and define few generalizations of these problems that appear naturally in the recursive steps. Let $\mathscr{F}$ be a family of sets over a universe A, then we define a bipartite graph $G_{\mathscr{F}}$ as follows. Let the bipartition of $V(G_{\mathscr{F}})$ be $A_{\mathscr{F}} \uplus B_{\mathscr{F}}$, where $A_{\mathscr{F}} = A$ and $B_{\mathscr{F}} = \mathscr{F}$. We define the edge set $E(G_{\mathscr{F}})$ as the set $\{\{u,Y\} \colon u \in A, u \in Y \in \mathscr{F}\}$. Let G be a C_4 free bipartite graph with bipartition $V(G) = A \uplus B$, and $\mathscr{F}$ be a family of sets over the universe A. We define the graph $G \cup G_{\mathscr{F}} = (A^* \uplus B^*, E^*)$ as follows. Let $A^* = A, B^* = B \uplus B_{\mathscr{F}}$ and $E^* = E(G) \cup E(G_{\mathscr{F}})$. The following problem generalizes HFVS on linear hypergraphs.

HITTING HYPERGRAPH FEEDBACK VERTEX SET (HHFVS) **Parameter:** $k + |E(H_2)|$
Input: Two linear hypergraphs H_1,H_2 such that $V(H_1) = V(H_2)$, $E(H_1) \cap E(H_2) = \emptyset$, and $H_1 \cup H_2$ is a linear hypergraph, $k \in \mathbb{N}$.
Question: Does there exist a set $S \subseteq V(H_1)$ of size at most k, such that $H_1 - S$ is acyclic and S is a hitting set for $E(H_2)$?

Observe that, HHFVS generalizes HFVS. Indeed, HHFVS is the same as HFVS, when H_2 is an empty hypergraph. Next, we define the "graph" version of HHFVS, which generalizes DFVSB on C_4-free graphs.

HITTING DOMINATING BIPARTITE FVS (HDBFVS)	**Parameter:** $k+\lvert\mathscr{F}\rvert$
Input: A C_4 free bipartite graph G with bipartition $V(G)=A\uplus B$, a family $\mathscr{F}$ of subsets of A such that the graph $G\cup G_{\mathscr{F}}$ is a C_4 free bipartite graph, $k\in\mathbb{N}$.	
Question: Does there exist a set $S\subseteq A$ of size at most k, such that $G-N[S]$ is a forest and S is a hitting set for $\mathscr{F}$?	

We say that an instance $(G=(A\uplus B,E),\mathscr{F},k)$ is a *valid instance* of HDBFVS, if $\mathscr{F}$ is a family of subsets of A such that the graph $G\cup G_{\mathscr{F}}$ is a C_4-free bipartite graph.

In the rest of the section, whenever we say $(G=(A\uplus B,E),\mathscr{F},k)$ is an instance of HDBFVS, it implies that $(G=(A\uplus B,E),\mathscr{F},k)$ is a valid instance of HDBFVS. Further, after each application of a reduction rule, we ensure that the instance remains valid.

Following is a simple observation, and its proof follows from the fact that $G\cup G_{\mathscr{F}}$ is C_4-free.

Observation 7.6.2. *If $(G=(A\uplus B,E),\mathscr{F},k)$ is an instance of* HDBFVS*, then (i) pairwise intersection of sets in $\mathscr{F}$ is of size at most* 1*, and (ii) for every vertex $b\in B$ and $F\in\mathscr{F}$, $\lvert N(b)\cap F\rvert$ is at most one.*

Given an instance (H_1,H_2,k) of HHFVS, we can obtain an instance, $(G,\mathscr{F},k)$, of HDBFVS in a canonical way. Next lemma shows their equivalence.

Lemma 49. *(H_1,H_2,k) is a* yes*-instance of* HHFVS *if and only if $(G,\mathscr{F},k)$ is a* yes*-instance of* HDBFVS*, where G is the incidence graph of the hypergraph H_1 and $\mathscr{F}=E(H_2)$.*

Proof. Observe that, $(G=(A\uplus B,E'),\mathscr{F},k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a solution to (H_1,H_2,k) of HHFVS. We claim that S is also a solution to $(G=(A\uplus B,E'),\mathscr{F},k)$ of HDBFVS. Suppose not. Then, either there exists a cycle $C=v_1e_1\ldots v_\ell e_\ell v_1$ in the graph $G-N_G[S]$ such that for each $i\in[\ell]$, $v_i\in A$, $e_i\in B$ and $v_ie_i\in E'$, and $e_\ell v_1\in E'$, or S does not hit a set $Y\in\mathscr{F}$. The former

case implies that, $e_1, \ldots, e_\ell$ are hyperedges in $H_1 - S$, and $\{v_1, \ldots, v_\ell\} \subseteq V(H_1) \setminus S$. Then, $(v_1, e_1, \ldots, v_\ell, e_\ell, v_1)$ is a cycle in the hypergraph $H_1 - S$. This is a contradiction to the assumption that $H_1 - S$ is acyclic. The later case implies that, there is an edge Y in $H_2 - S$, which is a contradiction to the assumption that $H_2 - S$ is edgeless (that is, S is a hitting set for H_2).

In the backward direction, let S' be a solution to $(G = (A \uplus B, E'), \mathscr{F}, k)$. We claim that S' is also a solution to (H_1, H_2, k) of HDBFVS. Suppose not. Then, either there exists a cycle $C = (v_1, e_1, \ldots, v_\ell, e_\ell, v_1)$ in the hypergraph $H_1 - S'$, or there exists an edge $Y \in H_2 - S$. The former case implies that, $\{v_1, \ldots, v_\ell\} \subseteq A \setminus S'$ and $\{e_1, \ldots, e_\ell\} \subseteq B \setminus N_G(S')$. Therefore, $v_1 e_1 \ldots v_\ell e_\ell v_1$ is a cycle in $G - N_G[S']$, which is a contradiction to the assumption that $G - N[S']$ is acyclic. The later case implies that, S' does not hit the set $Y \in \mathscr{F}$, a contradiction to the assumption that S' is a hitting set for $\mathscr{F}$. □

The rest of the section is devoted to designing an FPT algorithm for HDBFVS. Given an instance $(G = (A \uplus B, E), \mathscr{F}, k)$ of HDBFVS, we first define some notations that will be used throughout the section. For a vertex $v \in A$, by X_v we denote the set $\{Y \mid Y \in \mathscr{F}, v \in Y\}$. We *distinguish* the vertices in A as follows.

- If $|X_v| \geq 2$, that is, v is in at least two sets in $\mathscr{F}$, then we say that v is a *special* vertex.
- If $|X_v| = 1$, that is, v is contained in exactly one set in $\mathscr{F}$, then we say that v is an *easy* vertex.
- Otherwise, we say that v is a *trivial* vertex.

By $V(\mathscr{F})$ we denote the set $\{v \in A \mid v \in Y \text{ for some } Y \in \mathscr{F}\}$. For a graph $G^\star$, the notations $V_0(G^\star)$, $V_{=1}(G^\star)$, $V_{=2}(G^\star)$, and $V_{\geq 3}(G^\star)$ denote the set of isolated vertices, the set of vertices of degree 1, the set of vertices of degree 2, and the set of vertices of degree at least 3 in $G^\star$, respectively.

Lemma 50. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an instance of* HDBFVS. *Then, the number of special vertices in A is upper bounded by $\binom{|\mathscr{F}|}{2}$.*

Proof. For the sake of contradiction, assume that the number of special vertices in A is more than $\binom{|\mathscr{F}|}{2}$. Then, by pigeonhole principle there exist two special vertices u, v in A such that $|X_u \cap X_v| \geq 2$. Let $Y_1, Y_2 \in X_u \cap X_v$. This implies that $u, v \in Y_1 \cap Y_2$, which contradicts Observation 7.6.2(i). □

Now we state some reduction rules that are applied exhaustively by the algorithm in the order in which they appear. Let $(G, \mathscr{F}, k)$ be an instance of HDBFVS and $(G', \mathscr{F}', k)$ be the resultant instance after application of a reduction rule. To show that a reduction rule is safe, we will prove that $(G, \mathscr{F}, k)$ is a yes-instance if and only if $(G', \mathscr{F}', k)$ is a yes-instance.

Reduction Rule 27. *If one of the following holds, then return a trivial* no*-instance: (i)* $k < 0$*; (ii)* $k = 0$ *and* G *is not acyclic; and (ii)* $k = 0$ *and* $\mathscr{F}$ *is not empty.*

Reduction Rule 28. *If* $k \geq 0$*,* G *is acyclic and* $\mathscr{F}$ *is empty, then return a trivial* yes*-instance.*

Reduction Rule 29. *Let* $(G = (A \uplus B, E), \mathscr{F}, k)$ *be an instance of* HDBFVS *and* $b \in B$ *be a vertex that does not participate in any cycle in* G*. Then, output* $(G - b, \mathscr{F}, k)$*.*

Reduction Rule 30. *Let* $(G = (A \uplus B, E), \mathscr{F}, k)$ *be an instance of* HDBFVS *and* $v \in A$ *be an isolated vertex in* G*. If* v *is a trivial vertex, then output* $(G - v, \mathscr{F}, k)$*.*

It is easy to see that the above reduction rules are safe and can be applied in polynomial time. Observe that, when Reduction Rules 29 and 30 are no longer applicable, then $V_0(G) \subseteq A$ and each isolated vertex in G is either easy or special. Next, we state a reduction rule that will help to bound the number of easy isolated vertices in G.

Reduction Rule 31. *Let* $(G = (A \uplus B, E), \mathscr{F}, k)$ *be an instance of* HDBFVS *and* $v \in A$ *be an isolated vertex in* G*. Suppose* v *is an easy vertex,* $X_v = \{Y\}$*, and* $|Y| > 1$*. Then output* $(G', \mathscr{F}', k)$*, where* $G' = G - v$ *and* $\mathscr{F}' = (\mathscr{F} \setminus \{Y\}) \cup \{(Y \setminus \{v\})\}$*.*

Lemma 51. *Reduction Rule 31 is safe.*

Proof. Observe that, the instance $(G', \mathscr{F}', k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Observe that, if S does not contain v, then S is also a solution to $(G', \mathscr{F}', k)$, as $G' - N_{G'}[S] = (G - N_G[S]) - v$, $(G - N_G[S])$ is acyclic and S is also a hitting set of $\mathscr{F}'$. Next, consider the case when $v \in S$. Let $S' = S \setminus \{v\}$. Since v is an isolated vertex in G, we have that $G - N[S']$ is acyclic. Let $u \in Y$, $u \neq v$, then observe that, $S' \cup \{u\}$ is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS, which does not contain v and hence a solution to $(G, \mathscr{F}', k)$.

In the backward direction, let S' be a solution to $(G', \mathscr{F}', k)$. Suppose that, $G - N[S']$ contains a cycle C. Then, since $G' = G - v$, and $d_G(v) = 0$, C is also a cycle in $G' - N[S']$. Observe that, $\mathscr{F} \setminus \{Y\} = \mathscr{F}' \setminus (Y \setminus \{v\})$. Therefore, S' is also a hitting set of $\mathscr{F}$. This implies that S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. □

Reduction Rule 32. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an instance of* HDBFVS *and $v \in A$ be a vertex of degree* 1 *in G. If v is a trivial vertex, then output $(G', \mathscr{F}, k)$, where $G' = G - v$.*

Lemma 52. *Reduction Rule 32 is safe.*

Proof. Observe that, the instance $(G', \mathscr{F}, k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a solution to $(G, \mathscr{F}, k)$ of HDBFVS. If S does not contain v, then clearly S is also a solution to $(G', \mathscr{F}, k)$ because $G' - N_{G'}[S] = (G - N_G[S]) - v$ and $(G - N_G[S])$ is acyclic. Suppose that, $v \in S$. Let $\{b\} = N_G(v)$. Since Reduction Rule 29 is no longer applicable, we have that $d_G(b) > 1$. Let $u \neq v$ be an arbitrary vertex in $N_G(b)$. Then, $S^\star = (S \setminus \{v\}) \cup \{u\}$ is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS because $N_G(v) \subseteq N_G(u)$ and $d_G(v) = 1$. Then, $S^\star$ is also a solution to $(G' = G - v, \mathscr{F}, k)$ of HDBFVS because $G' - N_{G'}[S^\star] = (G - N_G[S^\star]) - v$ and $(G - N_G[S^\star])$ is acyclic.

In the backward direction, let S' be a solution to $(G', \mathscr{F}, k)$. Suppose that, $G - N[S']$ contains a cycle C. Then, since $G' = G - v$, and $d_G(v) = 1$, C is also a cycle in $G' - N[S']$. This implies that S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. □

When Reduction Rules 27 to 32 are no longer applicable, we obtain the following result.

Lemma 53. *Let $(G, \mathscr{F}, k)$ be an instance reduced with respect to Reduction Rules 27 to 32. Then, the following holds.*

1. *$V_0(G) \cup V_{=1}(G) \subseteq A$, all vertices in $V_0(G) \cup V_{=1}(G)$ are either easy or special.*
2. *$|V_0(G)| \leq |\mathscr{F}| + \binom{|\mathscr{F}|}{2}$.*

Lemma 54. *For any vertex $b \in B$, $|N_G(b) \cap V_{=1}(G)| \leq |\mathscr{F}|$.*

Proof. If there exists a vertex $v \in N_G(b) \cap V_{=1}(G)$ which is a trivial vertex, then Reduction Rule 32 is applicable. Thus, (i) for all $v \in N_G(b) \cap V_{=1}(G)$, v belongs to some set in $\mathscr{F}$. Suppose that, for a contradiction, there exists a vertex $b \in B$ such that $N_G(b)$ contains at least $|\mathscr{F}| + 1$ vertices of degree 1 in G. Then, by pigeonhole principle and statement (i), at least two degree 1 vertices say $u, v \in N_G(b)$ are contained in a set $Y \in \mathscr{F}$, which is a contradiction to item (ii) of Observation 7.6.2. This completes the proof of the lemma. □

Recall the definition of a degree two-path; P is a degree two path in G if each vertex in P has degree exactly two in G. Next, we state the reduction rules that help us bound the length of a long degree two path in $G - V_{=1}(G)$, that is, to bound the length of degree two paths in the graph obtained after deleting vertices of degree 1 from G. Towards this, we first define the notion of a *nice path*.

Definition 28. *We say that P is a* nice path *in G, if P does not have any* special *vertex, and the degree of each vertex in P in the graph $G - V_{=1}(G)$ is exactly* 2*. A* nice path *P in G is a* degree two nice path *if each vertex in P has degree exactly* 2 *in G.*

Reduction Rule 33. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an instance of* HDBFVS*, P be a nice path in G and $b, b' \in B$ be two vertices in P. If there exist two easy vertices u, u' whose degree is* 1 *in G, adjacent to b, b', respectively, such that $X_u = X_{u'} = \{Y\}$, then return $(G', \mathscr{F}', k)$, where $G' = G - u$, $\mathscr{F}' = (\mathscr{F} \setminus \{Y\}) \cup \{Y \setminus \{u\}\}$.*

Lemma 55. *Reduction Rule 33 is safe.*

Proof. Observe that, the instance $(G', \mathscr{F}', k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose that, $u \notin S$. Since $d_G(u) = 1$, we have that u does not participate in any cycle in G. Therefore, any cycle C in $G' - N[S]$ is also a cycle in $G - N[S]$. This implies that $G' - N[S]$ is acyclic. Observe that, $\mathscr{F} \setminus \{Y\} = \mathscr{F}' \setminus \{Y \setminus \{u\}\}$. This implies that S is a hitting set of $\mathscr{F}'$. Hence, S is also a solution to $(G', \mathscr{F}', k)$ of HDBFVS. Next, consider that $u \in S$. Since u does not participate in any cycle in G, u is only used to hit cycles containing b (recall that when we delete u, we also delete all its neighbors) and to hit the set Y. Since P is a nice path, any cycle that contains b also contains all the vertices in P and hence contains $N_G(u') = b'$, therefore u' can hit all the cycles containing b. Further, since $u' \in Y$, it holds that u' hits the set Y. This implies that $S^* = (S \setminus \{u\}) \cup \{u'\}$ is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. As argued before, S^* is a solution to $(G', \mathscr{F}', k)$ of HDBFVS.

In the backward direction, let S' be a solution to $(G', \mathscr{F}', k)$ of HDBFVS. Since u does not participate in any cycle, any cycle in $G - N[S']$ is also a cycle in $G' - N[S']$. Hence, $G - N[S']$ is acyclic. Also, since $\mathscr{F} \setminus \{Y\} = \mathscr{F}' \setminus \{Y \setminus \{u\}\}$, we have that S' is a hitting set of $\mathscr{F}$. Hence, S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. □

Lemma 56. *Let* $(G = (A \uplus B, E), \mathscr{F}, k)$ *be an instance of* HDBFVS *reduced with respect to Reduction Rules 27 to 33. Then, in any nice path P in G, the number of vertices that are adjacent to a vertex of degree* 1 *in G is bounded by* $\binom{|\mathscr{F}|}{2} + |\mathscr{F}|$.

Proof. From statement 1 in Lemma 53, we have that $V_{=1}(G) \subseteq A$. This implies, $N_G(V_{=1}(G)) \subseteq B$. Also, each vertex in $V_{=1}(G)$ is either easy or special. By Lemma 50, the number of vertices that are special is bounded by $\binom{|\mathscr{F}|}{2}$. Therefore, the number of vertices in P that are adjacent to special degree 1 vertices is at most $\binom{|\mathscr{F}|}{2}$. Since Reduction Rule 33 is no longer applicable, we have that corresponding to each set $Y \in \mathscr{F}$, there exists at most 1 vertex in P that has a degree 1 neighbor u such that $X_u = \{Y\}$. This implies that at most

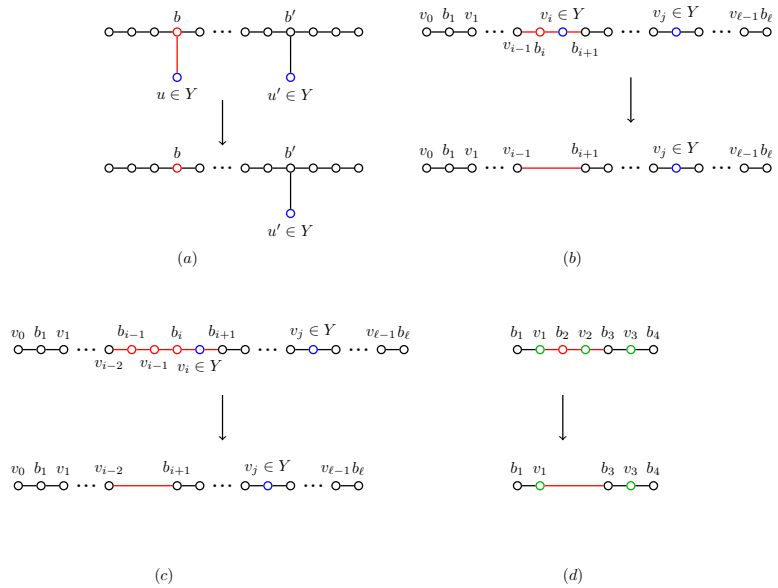

Figure 7.1: (a) is an illustration of Reduction Rule 33, (b) and (c) are illustrations of two cases of Reduction Rule 34, (d) is an illustration of Reduction Rule 35. In (a), (b) and (c) *blue* vertices denote *easy* vertices, and in (d) *green* vertices denote *trivial* vertices.

$|\mathscr{F}|$ vertices in P can be adjacent to degree 1 easy vertices, resulting in the mentioned upper bound. □

The next reduction rule helps us in upper bounding the length of degree two paths in G.

Reduction Rule 34. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an instance of* HDBFVS *and $P = v_0b_1v_1 \ldots v_{\ell-1}b_\ell$ be a degree two nice path in G, where $\{b_1, \ldots, b_\ell\} \subseteq B$, $\{v_0, \ldots, v_{\ell-1}\} \subseteq A$, and $\ell \geq 5$. Let $v_i, v_j \in A \cap (V(P) \setminus \{v_0, v_1\})$ be two distinct easy vertices such that $X_{v_i} = X_{v_j} = \{Y\}$ for some $Y \in \mathscr{F}$ and $i < j$.*

1. *If $X_{v_{i-1}} \neq X_{v_{i+1}}$ or $X_{v_{i-1}} = X_{v_{i+1}} = \emptyset$, then let $G' = (G - \{b_i, v_i\}) + v_{i-1}b_{i+1}$ (that is, G' be the graph obtained by deleting the vertices b_i, v_i from G and by adding a new edge $v_{i-1}b_{i+1}$) and $\mathscr{F}' = (\mathscr{F} \setminus \{Y\}) \cup \{Y \setminus \{v_i\}\}$.*

2. *Otherwise, $X_{v_{i-1}} = X_{v_{i+1}} = \{Y^\star\}$, then let $G' = (G - \{b_{i-1}, v_{i-1}, b_i, v_i\}) + v_{i-2}b_{i+1}$ (that is, G' be the graph obtained by deleting the vertices $b_{i-1}, v_{i-1}, b_i, v_i$ from G and by adding a new edge $v_{i-1}b_{i+1}$) and $\mathscr{F}' = (\mathscr{F} \setminus \{Y, Y^\star\}) \cup \{Y^\star \setminus \{v_{i-1}\}, Y \setminus \{v_i\}\}$.*

Return $(G', \mathscr{F}', k)$.

Lemma 57. *Reduction Rule 34 is safe.*

Proof. We first give a proof for Case 1, followed by a proof of Case 2.

Case 1: $X_{v_{i-1}} \neq X_{v_{i+1}}$ **or** $X_{v_{i-1}} = X_{v_{i+1}} = \emptyset$**.** The vertices v_{i-1} and b_{i+1} do not have two common neighbors in G', and hence there is no C_4 in G'. Observe that $G'_{\mathscr{F}'}$ is a subgraph of $G_{\mathscr{F}}$. Further, since $G_{\mathscr{F}}$ does not have C_4, $G'_{\mathscr{F}'}$ does not have C_4. Now we claim that there is no C_4 in $G' \cup G'_{\mathscr{F}'}$. There is no C_4 in G', $G'_{\mathscr{F}'}$, and $G \cup G_{\mathscr{F}}$. Thus, if there is a C_4 in $G' \cup G'_{\mathscr{F}'}$, then there is a set $F \in \mathscr{F}'$ such that $|(N_{G'}(b_{i+1}) \cap F| \geq 2$. Notice that $N_{G'}(b_{i+1}) = \{v_{i-1}, v_{i+1}\}$. Since ($X_{v_{i-1}} \neq X_{v_{i+1}}$ or $X_{v_{i-1}} = X_{v_{i+1}} = \emptyset$) and $|X_{v_{i-1}}|, |X_{v_{i+1}}| \leq 1$ (because P does not have any special vertex), there is no set $F \in \mathscr{F}'$ such that $\{v_{i-1}, v_{i+1}\} \subseteq F$. Thus, we have proved that there is no C_4 in $G' \cup G'_{\mathscr{F}'}$. This implies that the instance $(G', \mathscr{F}', k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose that, $v_i \notin S$. Then, we claim that S is also a solution to $(G', \mathscr{F}', k)$ of HDBFVS. Suppose not, then either there exists a cycle C in $G' - N_{G'}[S]$ or there exists a set $Z \in \mathscr{F}'$ such that $S \cap Z = \emptyset$. First consider the former case. If C does not contain the edge $v_{i-1}b_{i+1}$, then C is also a cycle in $G - N_G[S]$, which is a contradiction. Therefore, C contains the edge $v_{i-1}b_{i+1}$. But, then we get a cycle in $G - N_G[S]$ by replacing the edge $v_{i-1}b_{i+1}$ in C by the path $v_{i-1}b_iv_ib_{i+1}$. This is a contradiction to the assumption that $(G - N[S])$ is acyclic. Now, consider the later case. Note that S hits $\mathscr{F} \setminus \{Y\}$ and $Y \setminus \{v_i\}$ (since $v_i \notin S$). Thus, it implies that S is a hitting set of $\mathscr{F}'$. Hence, S is also a solution to $(G', \mathscr{F}', k)$ of HDBFVS. Next, consider that $v_i \in S$. Since P is a degree two nice path in G, any cycle that contains a vertex from $N[v_i]$ also contains all the vertices in P. In particular, it contains v_j, and v_j hits all the cycles that any vertex in $N[v_i]$ hits. Also, observe that, $v_j \in Y$ and hence v_j hits the set Y. This implies that $S^\star = S \setminus \{v_i\} \cup \{v_j\}$ is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. As argued before $S^\star$ is a solution to $(G', \mathscr{F}', k)$ of HDBFVS.

In the backward direction, let S' be a solution to $(G', \mathscr{F}', k)$ of HDBFVS. We claim that S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose not. Then, either there exists a cycle C in $G - N_G[S']$ or there exists a set $Z \in \mathscr{F}$ such that $S' \cap Z = \emptyset$. First consider the former case. If C does not contain any edge from the path P, then C is also a cycle in $G' - N_{G'}[S']$, which is a contradiction. Therefore, at least one edge from the path P is part of C. Then, since P is a degree two nice path in G, P is a subpath of C. Then, we get a cycle C' in $G' - N_{G'}[S']$ by replacing the subpath $v_{i-1}b_iv_ib_{i+1}$ in C by $v_{i-1}b_{i+1}$. This is a contradiction to the assumption that S' is a solution to $(G', \mathscr{F}', k)$. Now, consider the later case. Since $\mathscr{F} \setminus \{Y\} = \mathscr{F}' \setminus \{Y \setminus \{v_i\}\}$ and $|Y| \geq 2$, we have that S' is a hitting set of $\mathscr{F}$. Hence, S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS.

Case 2: $X_{v_{i-1}} = X_{v_{i+1}} = \{Y^*\}$. The vertices v_{i-2} and b_{i+1} do not have two common neighbors in G', and hence there is no C_4 in G'. Observe that $G'_{\mathscr{F}'}$ is a subgraph of $G_{\mathscr{F}}$. Further, since $G_{\mathscr{F}}$ does not have C_4, $G'_{\mathscr{F}'}$ does not have C_4. Next we claim that there is no C_4 in $G' \cup G'_{\mathscr{F}'}$. By item (ii) of Observation 7.6.2, $X_{v_{i-2}} \neq X_{v_{i-1}}$. This implies that $X_{v_{i-2}} \neq X_{v_{i+1}}$. Note that, there is no C_4 in G', $G'_{\mathscr{F}'}$, and $G \cup G_{\mathscr{F}}$. Thus, if there is a C_4 in $G' \cup G'_{\mathscr{F}'}$, then there exists a set $F \in \mathscr{F}'$ such that $|(N_{G'}(b_{i+1}) \cap F| \geq 2$. Notice that $N_{G'}(b_{i+1}) = \{v_{i-2}, v_{i+1}\}$. Since $X_{v_{i-2}} \neq X_{v_{i+1}}$ and $|X_{v_{i-2}}|, |X_{v_{i+1}}| \leq 1$ (because P does not have any special vertex), there is no set $F \in \mathscr{F}'$ such that $\{v_{i-2}, v_{i+1}\} \subseteq F$. Thus, we have proved that there is no C_4 in $G' \cup G'_{\mathscr{F}'}$. This implies that the instance $(G', \mathscr{F}', k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a minimal solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose $v_{i-1} \in S$ or $v_i \in S$. Consider the case when $v_{i-1} \in S$. Then, we claim that $S^\star = (S \setminus \{v_{i-1}\}) \cup \{v_{i+1}\}$ is also a solution to $(G, \mathscr{F}, k)$. Since P is a nice path, any cycle that contains a vertex of P must contain all the vertices of P. Thus, all the cycles containing a vertex from $N[v_{i-1}]$, also contain v_{i+1}. Therefore v_{i+1} hits all those cycles that $N[v_{i-1}]$ hits. Since $X_{v_{i-1}} = X_{v_{i+1}} = \{Y^\star\}$, v_{i+1} and v_{i-1} hits the same set (only one) from $\mathscr{F}$. Now suppose that $v_i \in S$. Then, we claim that $S' = (S \setminus \{v_i\}) \cup \{v_j\}$ is a solution to $(G, \mathscr{F}, k)$. Since all

the cycles containing a vertex from $N[v_i]$, also contain v_j, therefore v_j hits all the cycles that $N[v_i]$ hits. Since $X_{v_i} = X_{v_j} = \{Y^\star\}$, v_i and v_j hits the same set (only one) from $\mathscr{F}$.

Thus, if $(G, \mathscr{F}, k)$ is a yes-instance, then there is a solution S such that $v_{i-1}, v_i \notin S$. Then, we claim that S is also a solution to $(G', \mathscr{F}', k)$ of HDBFVS. Suppose not, then either there exists a cycle C in $G' - N_{G'}[S]$ or there exists a set $Z \in \mathscr{F}'$ such that $S \cap Z = \emptyset$. First consider the former case. If C does not contain the edge $v_{i-2}b_{i+1}$, then C is also a cycle in $G - N_G[S]$, which is a contradiction. Therefore, C contains the edge $v_{i-2}b_{i+1}$. But, then we get a cycle in $G - N_G[S]$ by replacing the edge $v_{i-2}b_{i+1}$ in C by the path $v_{i-2}b_{i-1}v_{i-1}b_iv_ib_{i+1}$. This is a contradiction to the assumption that $(G - N[S])$ is acyclic. Now, consider the later case. Note that S hits $\mathscr{F} \setminus \{Y, Y^\star\}$ and $\{Y^\star \setminus \{v_{i-1}\}, Y \setminus \{v_i\}\}$ (since $v_{i-1}, v_i \notin S$). Thus, it implies that S is a hitting set of $\mathscr{F}'$. Hence, S is also a solution to $(G', \mathscr{F}', k)$ of HDBFVS.

In the backward direction, let S' be a solution to $(G', \mathscr{F}', k)$ of HDBFVS. We claim that S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose not. Then, either there exists a cycle C in $G - N_G[S']$ or there exists a set $Z \in \mathscr{F}$ such that $S' \cap Z = \emptyset$. First consider the former case. If C does not contain any edge from the path P, then C is also a cycle in $G' - N_{G'}[S']$, which is a contradiction. Therefore, at least one edge from the path P is part of C. Since P is a degree two nice path in G, P is a subpath of C. Thus, we get a cycle C' in $G' - N_{G'}[S']$ by replacing the subpath $v_{i-2}b_{i-1}v_{i-1}b_iv_ib_{i+1}$ in C by $v_{i-2}b_{i+1}$. This is a contradiction to the assumption that S' is a solution to $(G', \mathscr{F}', k)$. Now, consider the later case. Since $\mathscr{F} \setminus \{Y, Y^\star\} = \mathscr{F}' \setminus \{Y^\star \setminus \{v_{i-1}\}, Y \setminus \{v_i\}\}$ and $|Y|, |Y^\star| \geq 2$, we have that S' is a hitting set of $\mathscr{F}$. Hence, S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. □

Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an instance of HDBFVS reduced with respect to Reduction Rules 27 to 34. Observe that, for each set $Y \in \mathscr{F}$ and a degree two nice path P in G, the number of easy vertices among the last $|V(P)| - 3$ vertices in $V(P)$ that belong to Y, is upper bounded by one. Reduction Rule 34 leads us to the following observation.

Observation 7.6.3. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be a reduced instance of* HDBFVS *with*

respect to Reduction Rules 27 to 34. Then, in any degree two nice path P of length at least 10 *in G, the number of easy vertices is bounded by $|\mathscr{F}|+2$.*

Reduction Rule 35. *Let $(G=(A \uplus B, E), \mathscr{F}, k)$ be an instance of* HDBFVS *and $P = b_1v_1b_2v_2b_3v_3b_4$ be a degree two nice path in G, such that $\{b_1,\ldots,b_4\} \subseteq B$, $\{v_1,v_2,v_3\} \subseteq A$ and v_1,v_2,v_3 are trivial vertices. Then, return $(G', \mathscr{F}, k)$, where G' is the graph obtained by deleting the vertices b_2, v_2 from G and adding a new edge v_1b_3 (that is, $G' = (G - \{v_2,b_2\}) + v_1b_3$).*

Lemma 58. *Reduction Rule 35 is safe.*

Proof. Observe that, the instance $(G', \mathscr{F}, k)$ is a valid instance of HDBFVS.

In the forward direction, let S be a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose that $v_2 \notin S$. Then, we claim that S is also a solution to $(G', \mathscr{F}, k)$ of HDBFVS. Suppose not, then there exists a cycle C in $G' - N_{G'}[S]$. If C does not contain the edge v_1b_3, then C is also a cycle in $G - N_G[S]$, which is a contradiction. Therefore, C contains the edge v_1b_3. But, then we get a cycle in $G - N_G[S]$ by replacing the edge v_1b_3 in C by the path $v_1b_2v_2b_3$. This is a contradiction to the assumption that $(G - N[S])$ is acyclic. Hence, S is also a solution to $(G', \mathscr{F}, k)$ of HDBFVS. Next, consider that $v_2 \in S$. Since P is a degree two nice path, any cycle that contains v_2 also contains all the vertices in P and hence contains v_1. Therefore $S^\star = S \setminus \{v_2\} \cup \{v_1\}$ is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. As argued before S^* is a solution to $(G', \mathscr{F}, k)$ of HDBFVS.

In the backward direction, let S' be a solution to $(G', \mathscr{F}, k)$ of HDBFVS. We claim that S' is also a solution to $(G, \mathscr{F}, k)$ of HDBFVS. Suppose not. Then, there exists a cycle C in $G - N_G[S']$. If C does not contain any edges from the path P, then C is also a cycle in $G' - N_{G'}[S']$, which is a contradiction. Therefore, at least one edge from the path P is part of C. Since P is a degree two nice path in G, P is a subpath in C. Thus, we get a cycle C' in $G' - N_{G'}[S']$ by replacing the subpath $v_1b_2v_2b_3$ in C by v_1b_3. This is a contradiction to the assumption that S' is a solution to $(G', \mathscr{F}, k)$. Hence, S' is also a solution to $(G, \mathscr{F}, k)$

of HDBFVS. □

Next, we have the following observation, which is easily verifiable.

Observation 7.6.4. *Let* $(G = (A \uplus B, E), \mathscr{F}, k)$ *be an instance of* HDBFVS *and let* $(G' = (A' \uplus B', E'), \mathscr{F}', k')$ *be the reduced instance of* HDBFVS *obtained from* $(G = (A \uplus B, E), \mathscr{F}, k)$*, by exhaustive applications of Reduction Rules 27 to 35. Then,* $|\mathscr{F}'| = |\mathscr{F}|$ *and* $k' \leq k$*.*

We now bound the size of degree 2 path, when there is no degree 1 vertex in the graph.

Lemma 59. *Let* $(G = (A \uplus B, E), \mathscr{F}, k)$ *be an instance of* HDBFVS *reduced with respect to Reduction Rules 27 to 35. Then, the number of vertices in a degree two path* P *in* $G - V_{=1}(G)$ *is bounded by* $63|\mathscr{F}|^5 + 21$*.*

Proof. By Lemma 50, the number of special vertices in P is bounded by $\binom{|\mathscr{F}|}{2}$. Let P' be a maximum length subpath of P such that P' is a nice path. That is, P' does not contain any special vertices. Then, by Lemma 56, the number of vertices in P' that are adjacent to a vertex in $V_{=1}(G)$ in G is bounded by $\binom{|\mathscr{F}|}{2} + |\mathscr{F}|$. Let P'' be a maximum length subpath of P' such that P'' does not contain any vertex that is adjacent to a vertex in $V_{=1}(G)$ in G. Then, by Observation 7.6.3, either the length of P'' is bounded by 10, or the number of easy vertices in P'' is bounded by $|\mathscr{F}| + 2$. Let $P^\star$ be a maximum length subpath of P'' such that $P^\star$ does not contain any easy vertices. Then, since Reduction Rule 35 is no longer applicable, the length of $P^\star$ is bounded by 7. Therefore, we have that the length of P'' is bounded by $7(|\mathscr{F}| + 3)$. This implies that the length of P' is bounded by $7(|\mathscr{F}| + 3)(\binom{|\mathscr{F}|}{2} + |\mathscr{F}| + 1) \leq (35|\mathscr{F}|^3 + 21)$. Hence, the length of P is bounded by $(35|\mathscr{F}|^3 + 21)(\binom{\mathscr{F}}{2} + 1) \leq 63|\mathscr{F}|^5 + 21$. □

From now on, we say that $(G = (A \uplus B, E), \mathscr{F}, k)$ is a *reduced instance* of HDBFVS if it is reduced with respect to Reduction Rules 27 to 35. In the following lemma, we observe

that, if $(G = (A \uplus B, E), \mathscr{F}, k)$ is a yes-instance of HDBFVS, then a large number of edges in G is incident to the neighborhood of the solution.

Lemma 60. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be a reduced instance of* HDBFVS *where G is not a forest. Then, for any solution S, at least $1/(445|\mathscr{F}|^6 + 68)$ fraction of the total edges in E are incident to $N[S]$.*

Proof. Let E_S be the set of edges incident to all the vertices of $N[S]$ in G. Observe that, $E(G) = E_S \uplus E(G - N[S])$. Since $G - N[S]$ is a forest, we have that $|E(G - N[S])| < |V(G - (N[S] \cup V_0(G)))|$. We aim to show that $|V(G - (N[S] \cup V_0(G)))| \leq (445|\mathscr{F}|^6 + 67) \cdot |E_S|$. Let $V^\star$ be the set of vertices of degree 1 in $G - N[S]$. Let $V_1^\star \subseteq V^\star$ be the set of vertices that have some neighbor in $N[S]$ and $V_2^\star = V^\star \setminus V_1^\star$. That is, $V_2^\star \subseteq V_{=1}(G)$. Since the vertices in $V_1^\star$ have neighbors in $N[S]$, they contribute at least one edge to the set E_S and these edges are distinct. Hence, $|V_1^\star| \leq |E_S|$.

Since $V_2^\star \subseteq V_{=1}(G)$, by Lemma 53, we have that $V_2^\star \subseteq A$. Thus, $V_2^\star$ have neighbors only in the set $B \cap V(G - N[S])$. Also, by Lemma 54, any vertex in B can be adjacent to at most $|\mathscr{F}|$ vertices of degree 1 in G. Hence, each vertex in $B \cap V(G - N[S])$ can be adjacent to at most $|\mathscr{F}|$ vertices of $V_2^\star$. Thus, we have that $|V_2^\star| \leq |\mathscr{F}| \cdot |B \cap V(G - N[S])|$. Let G' be the graph $G - (V_0(G) \cup V_2^\star)$. Since $V_0(G) \cup V_2^\star \subseteq A$, we have that, $B \subseteq V(G')$ and $B \cap V(G - N[S]) = B \cap V(G' - N[S])$. Hence, we obtain the following.

$$
\begin{aligned}
(7.1)\ |V_2^\star| &\leq |\mathscr{F}| \cdot |B \cap V(G' - N[S])| \leq |\mathscr{F}| \cdot |V(G' - N[S])| \\
(7.2)\ |V^\star| &= |V_1^\star| + |V_2^\star| \leq |\mathscr{F}| \cdot |V(G' - N[S])| + |E_S| \qquad \text{(By (7.1) and } |V_1^\star| \leq |E_S|\text{)}
\end{aligned}
$$

Since the graph G' is obtained from G by deleting a subset of vertices that are contained in $V_0(G) \cup V_{=1}(G) \subseteq A$, the vertices that are degree 1 in $G' - N[S]$ are either degree 1 vertices in $G - N[S]$ and are contained in A, in particular in $V_1^\star$, or they are contained in B and are neighbors of vertices in $V_2^\star$ in G. Let L be the set of leaves (vertices of degree 1) in $G' - N[S]$. We claim that $L = V_1^\star$. For contradiction, assume that a vertex $b \in B \cap L$. Since

Reduction Rule 29 is no longer applicable, we have that each vertex in B participates in a cycle in G and hence, participates in a cycle in G'. Therefore, degree of b is at least 2 in G'. Observe that b cannot have a neighbor in S, otherwise $b \in N[S]$. This implies that b has 2 neighbors in $G' - N[S]$, which contradicts that $b \in L$. Observe that each vertex in $V_1^\star$ is a leaf vertex in $G' - N[S]$. Hence $L = V_1^\star$. Therefore, we obtain the following.

$$|L| \leq |E_S|. \tag{7.3}$$

$$|V_{\geq 3}(G' - N[S])| \leq |E_S| \quad \text{(Since, } G' - N[S] \text{ is a forest, } V_{\geq 3}(G' - N[S]) \leq |L|) \tag{7.4}$$

Next we bound $|V_0(G' - N[S])|$. Since, for any vertex v in $G' - N[S]$, $d_G(v) \geq 1$, we have that any vertex $w \in V_0(G' - N[S])$ is adjacent to some vertex in $N[S]$. Then, each vertex in $V_0(G' - N[S])$ contributes at least 1 edge to the set E_S and these edges are distinct. Therefore, we obtain the following.

$$|V_0(G' - N[S])| \leq |E_S|. \tag{7.5}$$

Let $V^1_{=2}(G')$ be the set of vertices of degree 2 in $G' - N[S]$ that have a neighbor in $N[S]$. Then, each vertex in $V^1_{=2}(G')$ contributes at least 1 edge to the set E_S. Therefore, we obtain the following.

$$|V^1_{=2}(G')| \leq |E_S|. \tag{7.6}$$

Let $V^2_{=2}(G')$ be the set of vertices of degree 2 in $G' - N[S]$, that do not have a neighbor in $N[S]$. Then, each vertex in $V^2_{=2}(G')$ is contained in some maximal degree two path not containing any vertex of $V^1_{=2}(G')$ in $G' - N[S]$. Observe that, since $G' - N[S]$ is a forest, (i) the number of maximal degree two paths not containing any vertex of $V^1_{=2}(G')$ in $G' - N[S]$ is bounded by $|L \cup V_{\geq 3}(G') \cup V^1_{=2}(G')|$ and hence bounded by $3|E_S|$ (because of (7.3),(7.4), and (7.6)). Observe that a degree two path not containing any vertex of $V^1_{=2}(G')$

in $G' - N[S]$ is also a degree two path in $G - V_{=1}(G)$. By Lemma 59, (ii) the number of vertices in a degree two path in $G - V_{=1}(G)$ is bounded by $63|\mathscr{F}|^5 + 21$. Therefore, statements (i) and (ii) imply the following.

$$|V^2_{=2}(G')| \leq (189|\mathscr{F}|^5 + 63)|E_S| \tag{7.7}$$

Observe that $V_{=2}(G' - N[S]) = V^1_{=2}(G') \cup V^2_{=2}(G')$. By (7.6) and (7.7), we get the following.

$$|V_{=2}(G' - N[S])| = |V^1_{=2}(G')| + |V^2_{=2}(G')| \leq (189|\mathscr{F}|^5 + 64)|E_S| \tag{7.8}$$

Note that, $V(G' - N[S]) = V_0(G' - N[S]) \cup L \cup V_{\geq 3}(G' - N[S]) \cup V_{=2}(G' - N[S])$. Hence, we obtain the following using (7.3), (7.5), (7.4), and (7.8).

$$\begin{aligned} |V(G' - N[S])| &= |V_0(G' - N[S])| + |L| + |V_{\geq 3}(G' - N[S])| + |V_{=2}(G' - N[S])| \\ &\leq |E_S| + |E_S| + |E_S| + (189|\mathscr{F}|^5 + 64)|E_S| \\ &\leq (189|\mathscr{F}|^5 + 67)|E_S| \end{aligned} \tag{7.9}$$

Using (7.1) and (7.9), we obtain the following.

$$\begin{aligned} |V(G - (N[S] \cup V_0(G)))| &\leq |V(G' - N[S])| + |V^\star_2| \\ &\leq (|\mathscr{F}| + 1)|V(G' - N[S])| \qquad \text{(By (7.1))} \\ &\leq (|\mathscr{F}| + 1)((189|\mathscr{F}|^5 + 67)|E_S|) \\ &\leq (445|\mathscr{F}|^6 + 67)|E_S| \end{aligned}$$

Thus,
$$\begin{aligned} |E(G)| &= |E_S| + |E(G - N[S])| \\ &\leq |E_S| + |V(G - (N[S] \cup V_0(G))| \leq (445|\mathscr{F}|^6 + 68)|E_S|. \end{aligned}$$

This concludes the proof. □

Lemma 61. *Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an instance of* HDBFVS*, where G is a forest*

and $|\mathscr{F}| \leq k^2$. *Then, there exists an algorithm which solves the instance in* $\mathscr{O}^\star((2k^4)^k)$ *time.*

Proof. The Algorithm first applies Reduction Rules 27 to 35 exhaustively in the order in which they are stated. If any reduction rule solves the instance, then output yes and no accordingly. All the reduction rules are safe, and can be applied in polynomial time, and they can be applied only polynomial many times since each reduction rule decreases the size of the graph. Let $(G' = (A' \uplus B', E'), \mathscr{F}', k')$ be the reduced instance. Since Reduction Rule 29 is no longer applicable, $B' = \emptyset$, and hence G' is an edge-less graph with vertex set A'. By Lemma 53, $|V(G')| = |A'| \leq |\mathscr{F}'| + \binom{|\mathscr{F}'|}{2}$. By Observation 7.6.4, we have that $|\mathscr{F}'| = |\mathscr{F}| \leq k^2$ and hence, $|V(G')| \leq 2k^4$. We enumerate all the subsets of $V(G')$ of size at most k and check if either of them forms a solution; else return that it is a no-instance. Clearly, this algorithm runs in time $\binom{2k^4}{k} n^{\mathscr{O}(1)} = \mathscr{O}^\star((2k^4)^k)$. This completes the proof. □

Lemma 62. *There is a randomized algorithm that takes an instance* $(G = (A \uplus B, E), \mathscr{F}, k)$ *of* HDBFVS *as input, runs in* $\mathscr{O}^*((2k^4)^k)$ *time, and outputs either* yes*, or* no*, or an instance* $(G^\star = (A^\star \uplus B^\star, E^\star), \mathscr{F}^\star, k^\star)$ *of* HDBFVS *where* $k^\star < k$*, with the following guarantee.*

- *If* $(G, \mathscr{F}, k)$ *is a* yes*-instance, then the output is* yes *or an equivalent* yes*-instance* $(G^\star, \mathscr{F}^\star, k^\star)$ *where* $k^\star < k$*, with probability at least* $(445k^{12} + 68)^{-(k^2+1)}$.
- *If* $(G, \mathscr{F}, k)$ *is a* no*-instance, then the output is* no *or an equivalent* no*-instance* $(G^\star, \mathscr{F}^\star, k^\star)$ *where* $k^\star < k$*, with probability* 1.

Proof. Let $(G = (A \uplus B, E), \mathscr{F}, k)$ be an input instance of HDBFVS. Recall that, for any $v \in A$, $X_v = \{F \in \mathscr{F} : v \in F\}$. The algorithm applies the following iterative procedure.

Step 1. If G is acyclic and $|\mathscr{F}| \leq k^2$, then apply Lemma 61 and solve the instance.

Step 2. If $|\mathscr{F}| \geq k^2 + 1$;

(i) If there exists a vertex v such that $|X_v| \geq k+1$, return $(G - N[v], \mathscr{F} \setminus X_v, k-1)$.

(ii) Otherwise, return that $(G = (A \uplus B, E), \mathscr{F}, k)$ is a no-instance of HDBFVS.

Step 3. Apply Reduction Rules 27 to 35 exhaustively in the order in which they are stated. If any reduction rule solves the instance, then output yes and no accordingly. Let $(G' = (A' \uplus B', E'), \mathscr{F}', k')$ be the reduced instance.

Step 4. Pick an edge $e = ub$ in $E(G')$ uniformly at random, where $u \in A', b \in B'$. Set $G := G' - b, \mathscr{F} := \mathscr{F}' \cup \{N_{G'}(b)\}$, and $k := k'$. Go to Step 1.

Now we prove the correctness of the algorithm. Correctness of Step 1 follows from Lemma 61. Next assume that $|\mathscr{F}| \geq k^2 + 1$. Suppose that, v is a vertex that is contained in at least $k+1$ sets in $\mathscr{F}$. By Observation 7.6.2, pairwise intersection of two sets in $\mathscr{F}$ is at most 1. Thus, if we do not pick v in our solution, then we have to pick at least $k+1$ vertices to hit sets in X_v. This implies that v is contained in every solution of $(G, \mathscr{F}, k)$ of HDBFVS. Therefore, we have that $(G, \mathscr{F}, k)$ is a yes-instance of HDBFVS if and only if $(G - v, \mathscr{F} \setminus X_v, k-1)$ is a yes-instance of HDBFVS, and correctness of Step 2i follows. Suppose that, each vertex in A is contained in at most k sets in $\mathscr{F}$. This implies that no set of size at most k can hit k^2+1 sets of $\mathscr{F}$. Therefore, $(G, \mathscr{F}, k)$ is a no-instance of HDBFVS, and correctness of Step 2ii follows. Correctness of the Step 3 is implied by the fact that all our reduction rules are safe. Suppose that, the algorithm does not stop in Step 3. Let $(G', \mathscr{F}', k')$ be the reduced instance, where $k' \leq k$. Now, let S be a hypothetical solution to $(G', \mathscr{F}', k')$. By Lemma 60, the picked edge $e = ub$ is incident to a vertex in $N_{G'}[S]$ with probability at least $1/(445|\mathscr{F}|^6 + 68)$. This implies that with probability at least $1/(445|\mathscr{F}|^6 + 68)$ a vertex in $N_{G'}(b)$ is contained in S. Therefore, if $(G', \mathscr{F}', k')$ is a yes-instance, then $(G' - b, \mathscr{F}' \cup \{N_G(b)\}, k')$ is a yes-instance, with probability at least $1/(445|\mathscr{F}|^6 + 68)$. Also, notice that any solution to $(G' - b, \mathscr{F}' \cup \{N_G(b)\}, k')$ is also a solution to $(G', \mathscr{F}', k')$. Therefore, if $(G', \mathscr{F}', k')$ is a no-instance, then $(G' - b, \mathscr{F}' \cup \{N_G(b)\}, k')$ is a no-instance, with probability 1. Consequently,

if $(G,\mathscr{F},k)$ is a no-instance, then the output is no or a no-instance $(G^\star,\mathscr{F}^\star,k^\star)$ with probability 1.

Now, suppose that $(G,\mathscr{F},k)$ is yes-instance. By Observation 7.6.4, we know that after the application of Reduction Rules 27 to 35, (a) the size of the family $\mathscr{F}'$ in the reduced instance is $|\mathscr{F}|$. Therefore, Step 4 is applied at most k^2+1 times. Moreover, each execution of Step 4 is a *success* with probability at least $1/(445|\widehat{\mathscr{F}}|^6+68)$, where $\widehat{\mathscr{F}}$ is the family in the instance considered in that step. In Step 4, the size of the family of any instance is bounded by k^2, because of Step 2. Therefore each execution of Step 4 is a success with probability at least $1/(445k^{12}+68)$. This implies that either our algorithm outputs yes or a yes-instance $(G^\star,\mathscr{F}^\star,k^\star)$ with probability at least $(445k^{12}+68)^{-(k^2+1)}$. By Observation 7.6.4, we know that after the application of Reduction Rules 27 to 35, the parameter k' in the reduced instance is at most the parameter k in the original instance. Moreover, if the algorithm outputs an instance, then that will happen in Step 2i and there k decreases by 1. This implies that $k^\star < k$. This completes the proof of correctness of the algorithm.

By Lemma 61, Step 1 runs in $\mathscr{O}^\star((2k^4)^k)$ time. Observe that, Step 2 runs in polynomial time. All the reduction rules run in polynomial time, and are applied only polynomially many times. Step 4 runs in polynomial time, and we have at most k^2+1 iterations. Therefore, the total running time is $\mathscr{O}^\star((2k^4)^k)$. This completes the proof. □

By applying Lemma 62 at most k times, we can show the the following.

Lemma 63. *There exists a randomized algorithm $\mathscr{B}$ that takes an instance $(G=(A\uplus B,E),\mathscr{F},k)$ of* HDBFVS *as input, runs in $\mathscr{O}^\star((2k^4)^k)$ time, and outputs either* yes *or* no *with the following guarantee.*

- *If $(G,\mathscr{F},k)$ is a* yes*-instance, then the output is* yes *with probability at least $(445k^{12}+68)^{-k(k^2+1)}$.*
- *If $(G,\mathscr{F},k)$ is a* no*-instance, then the output is* no *with probability* 1.

Let $\tau(k) = (445k^{12}+68)^{k(k^2+1)}(\log n)^{\mathscr{O}(1)}$. To boost the success probability of algorithm $\mathscr{B}$, we repeat it $\mathscr{O}(\tau(k)\log n)$ times. After applying algorithm $\mathscr{B}$ $\mathscr{O}(\tau(k)\log n)$ times, the success probability is at least

$$1-\left(1-\frac{1}{\tau(k)}\right)^{\mathscr{O}(\tau(k)\log n)} \geq 1-\frac{1}{2^{\mathscr{O}(\log n)}} \geq 1-\frac{1}{n^{\mathscr{O}(1)}}.$$

Thus, we have the following result.

Theorem 7.6.5. *There exists a randomized algorithm $\mathscr{A}$ that takes an instance $(G=(A\uplus B,E),\mathscr{F},k)$ of* HDBFVS *as input, runs in $\mathscr{O}^{\star}(2^{\mathscr{O}(k^3\log k)})$ time, and outputs either* yes *or* no *with the following guarantee.*

- *If $(G,\mathscr{F},k)$ is a* yes*-instance, then the output is* yes *with probability at least $1-\frac{1}{n^{\mathscr{O}(1)}}$.*
- *If $(G,\mathscr{F},k)$ is a* no*-instance, then the output is* no *with probability* 1.

7.7 Conclusion

In this chapter, we initiated the study of FEEDBACK VERTEX SET problem on hypergraphs. We showed that the problem is W[2]-hard on general hypergraphs. However, when the input is restricted to d-hypergraphs and linear hypergraphs, which are a strict generalization of graphs, FVS turns out to be tractable (FPT). We believe that this opens up a new direction in the study of parameterized algorithms. That is, extending the study of other graph problems, in the realm of Parameterized Complexity, to hypergraphs. Designing substantially faster algorithms for HFVS on linear hypergraphs and designing polynomial kernels remain interesting questions for the future.

Bibliography

[1] Faisal N. Abu-Khzam. A kernelization algorithm for d-hitting set. *Journal of Computer and System Sciences*, 76(7):524–531, 2010.

[2] Akanksha Agrawal, Sushmita Gupta, Saket Saurabh, and Roohani Sharma. Improved algorithms and combinatorial bounds for independent feedback vertex set. In *11th International Symposium on Parameterized and Exact Computation IPEC*, volume 63, pages 2:1–2:14, 2016.

[3] Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In *Theoretical Informatics - 12th Latin American Symposium LATIN*, volume 9644 of *LNCS*, pages 1–13. Springer, 2016.

[4] Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. Simultaneous feedback vertex set: A parameterized perspective. *ACM Transactions on Computation Theory*, 10(4):18:1–18:25, 2018.

[5] Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. Choice is hard. In *26th International Symposium on Algorithms and Computation ISAAC*, pages 318–328, 2015.

[6] Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. Conflict-free covering. In *27th Canadian Conference on Computational Geometry CCCG*, 2015.

[7] Aritra Banik, Fahad Panolan, Venkatesh Raman, and Vibha Sahlot. Fréchet distance between a line and avatar point set. *Algorithmica*, 80(9):2616–2636, 2018.

[8] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research*, 12:219–234, 2000.

[9] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[10] Matthias Bentert, René van Bevern, and Rolf Niedermeier. (wireless) scheduling, graph classes, and c-colorable subgraphs. *CoRR*, abs/1712.06481, 2017.

[11] Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, (049), 2003.

[12] Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[13] Hans L. Bodlaender. On disjoint cycles. In Gunther Schmidt and Rudolf Berghammer, editors, *Proceedings on Graph–Theoretic Concepts in Computer Science (WG '91)*, volume 570 of *LNCS*, pages 230–238. Springer, 1992.

[14] Hans L. Bodlaender. A cubic kernel for feedback vertex set. In *24th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, volume 4393 of *LNCS*, pages 320–331. Springer, 2007.

[15] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

[16] Béla Bollobás. *Extremal graph theory*. Courier Corporation, 2004.

[17] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 2013.

[18] 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 *Parameterized and Exact Computation, Second International Workshop, IWPEC*, volume 4169 of *LNCS*, pages 192–202. Springer, 2006.

[19] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[20] Leizhen Cai and Junjie Ye. Dual connectedness of edge-bicolored graphs and beyond. In *International Symposium on Mathematical Foundations of Computer Science*, volume 8635, pages 141–152, 2014.

[21] Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the 27th annual ACM-SIAM symposium on Discrete algorithms SODA*, pages 1096–1115, 2016.

[22] Yixin Cao. A naive algorithm for feedback vertex set. In *1st Symposium on Simplicity in Algorithms, SOSA*, volume 61, pages 1:1–1:9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[23] Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.

[24] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015.

[25] Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016.

[26] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188 – 1198, 2008.

[27] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[28] Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *CoRR*, abs/1205.1271, 2012.

[29] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer-Verlag, 2015.

[30] Marek Cygan, Fabrizio Grandoni, and Danny Hermelin. Tight kernel bounds for problems on graphs with small degeneracy. *ACM Transactions on Algorithms*, 13(3):43:1–43:22, 2017.

[31] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 150–159. IEEE Computer Society, 2011.

[32] Marek Cygan and Marcin Pilipczuk. Split vertex deletion meets vertex cover: New fixed-parameter and exact exponential-time algorithms. *Information Processing Letters*, 113(5-6):179–182, 2013.

[33] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM Journal on Discrete Mathematics*, 27(1):290–309, 2013.

[34] G Dantzig and Delbert Ray Fulkerson. On the max flow min cut theorem of networks. *Linear inequalities and related systems*, 38:225–231, 2003.

[35] Andreas Darmann, Ulrich Pferschy, and Joachim Schauer. Determining a minimum spanning tree with disjunctive constraints. In *International Conference on Algorithmic Decision Theory ADT*, volume 5783 of *LNCS*, pages 414–423. Springer, 2009.

[36] 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.

[37] Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory of Computing Systems*, 41(3):479–492, 2007.

[38] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *Journal of the ACM*, 61(4):23:1–23:27, 2014.

[39] Zhuo Diao and Zhongzheng Tang. On the feedback number of 3-uniform hypergraph. volume abs/1807.10456, 2018.

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

[41] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[42] Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier, and Anke Truß. Fixed-parameter tractability results for feedback set problems in tournaments. *Journal of Discrete Algorithms*, 8(1):76–86, 2010.

[43] Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. *Complexity Theory: Current Research*, pages 191–225, 1992.

[44] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[45] Leah Epstein, Lene M. Favrholdt, and Asaf Levin. Online variable-sized bin packing with conflicts. *Discrete Optimization*, 8(2):333–343, 2011.

[46] Guy Even, Magnús M. Halldórsson, Lotem Kaplan, and Dana Ron. Scheduling with conflicts: online and offline algorithms. *Journal of Scheduling*, 12(2):199–224, 2009.

[47] Shimon Even and Oded Kariv. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. *Foundations of Computer Science (FOCS)*, pages 100–112, 1975.

[48] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009.

[49] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[50] Stéphane Foldes and Peter L Hammer. *Split graphs*. Universität Bonn. Institut für Ökonometrie und Operations Research, 1976.

[51] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *IEEE 53rd Annual Symposium on Foundations of Computer Science FOCS*, pages 470–479, 2012.

[52] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM*, 63(4):29:1–29:60, 2016.

[53] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

[54] Toshihiro Fujito. A unified approximation algorithm for node-deletion problems. *Discrete Applied Mathematics*, 86:213–231, 1998.

[55] Toshihiro Fujito. Approximating minimum feedback vertex sets in hypergraphs. *Theoretical Computer Science*, 246(1):107 – 116, 2000.

[56] Zoltán Füredi. On the number of edges of quadrilateral-free graphs. *Journal of Combinatorial Theory, Series B*, 68(1):1–6, 1996.

[57] Harold N. Gabow, Shachindra N Maheshwari, and Leon J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, 2(3):227–231, 1976.

[58] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[59] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.

[60] Dan Gusfield and Leonard Pitt. A bounded approximation for the minimum cost 2-sat problem. *Algorithmica*, 8(2):103–117, 1992.

[61] Yoichi Iwata and Yusuke Kobayashi. Improved analysis of highest-degree branching for feedback vertex set. In *14th International Symposium on Parameterized and Exact Computation, IPEC*, pages 22:1–22:11, 2019.

[62] Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time fpt algorithms via network flow. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1749–1761. SIAM, 2014.

[63] Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM Journal on Computing*, 45(4):1377–1411, 2016.

[64] Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/all CSPs, half-integral a-path packing, and linear-time FPT algorithms. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 462–473. IEEE Computer Society, 2018.

[65] Minghui Jiang. On the parameterized complexity of some optimization problems related to multiple-interval graphs. *Theoretical Computer Science*, 411(49):4253–4262, 2010.

[66] Viggo Kann. *Polynomially bounded minimization problems which are hard to approximate*, pages 52–63. Springer Berlin Heidelberg, 1993.

[67] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. *Journal of Combinatorial Theory, Series B*, 102(4):1020–1034, 2012.

[68] Ken-ichi Kawarabayashi and Bruce Reed. An (almost) linear time algorithm for odd cycles transversal. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 365–378. SIAM, 2010.

[69] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.

[70] Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.

[71] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20:1–20:15, 2014.

[72] KW Krause, MA Goodwin, and RW Smith. *Optimal software test planning through automated network analysis*. TRW Systems Group, 1973.

[73] Melven R Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.

[74] Mithilesh Kumar and Daniel Lokshtanov. Faster exact and parameterized algorithm for feedback vertex set in tournaments. In *33rd Symposium on Theoretical Aspects of Computer Science STACS*, volume 47, pages 49:1–49:13, 2016.

[75] Stefan Langerman and Pat Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005.

[76] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[77] Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^*(2.7^k)$ time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms SODA*, pages 971–989. SIAM, 2020.

[78] Shaohua Li and Marcin Pilipczuk. An improved FPT algorithm for independent feedback vertex set. In *44th International Workshop on Graph-Theoretic Concepts in Computer Science WG*, volume 11159, pages 344–355. Springer, 2018.

[79] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 922–934, 2015.

[80] Daniel Lokshtanov, NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms (TALG)*, 11(2):1–31, 2014.

[81] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering small independent sets and separators with applications to parameterized algorithms. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2785–2800, 2018.

[82] Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. Linear time parameterized algorithms for subset feedback vertex set. *ACM Transactions on Algorithms*, 14(1):7:1–7:37, 2018.

[83] Daniel Lokshtanov, MS Ramanujan, and Saket Saurabh. When recursion is better than iteration: a linear-time algorithm for acyclicity with few error vertices. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms SODA*, pages 1916–1933. SIAM, 2018.

[84] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994.

[85] Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009.

[86] Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.

[87] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 30(3):417–427, 1983.

[88] Silvio Micali and Vijay V Vazirani. An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. *Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[89] Neeldhara Misra, N. S. Narayanaswamy, Venkatesh Raman, and Bal Sri Shankar. Solving min ones 2-sat as fast as vertex cover. *Theoretical Computer Science*, 506:115–121, 2013.

[90] Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On parameterized independent feedback vertex set. *Theoretical Computer Science*, 461:65–75, 2012.

[91] Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. FPT algorithms for connected feedback vertex set. *Journal of Combinatorial Optimization*, 24(2):131–146, 2012.

[92] Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms for even cycle transversal. In *Graph-Theoretic Concepts in Computer Science - 38th International Workshop, WG*, volume 7551 of *LNCS*, pages 172–183. Springer, 2012.

[93] Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. *Foundations of Computer Science (FOCS)*, pages 182–191, 1995.

[94] Jaroslav Nešetřil and Patrice Ossona de Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

[95] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

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

[97] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, 2006.

[98] Ulrich Pferschy and Joachim Schauer. The maximum flow problem with conflict and forcing conditions. In *International Conference on Network Optimization INOC*, volume 6701 of *LNCS*, pages 289–294. Springer, 2011.

[99] Ulrich Pferschy and Joachim Schauer. The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26(1):109–119, 2013.

[100] Ulrich Pferschy and Joachim Schauer. Approximation of knapsack problems with conflict and forcing graphs. *Journal of Combinatorial Optimization*, 33(4):1300–1323, 2017.

[101] Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Transactions on Algorithms*, 9(1):11:1–11:23, 2012.

[102] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[103] Jan Arne Telle and Yngve Villanger. FPT algorithms for domination in biclique-free graphs. In *ESA*, pages 802–812, 2012.

[104] Jan Arne Telle and Yngve Villanger. FPT algorithms for domination in sparse graphs and beyond. *Theoretical Computer Science*, 770:62–68, 2019.

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

[106] René van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015.

[107] Magnus Wahlström. *Algorithms, measures and upper bounds for satisfiability and related problems*. PhD thesis, Linköping University, Sweden, 2007.

[108] L Wang, EK Egorova, and AV Mokryakov. Development of hypergraph theory. *Journal of Computer and Systems Sciences International*, 57(1):109–114, 2018.

[109] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. *Symposium on Theory of Computing (STOC)*, pages 887–898, 2012.

[110] Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 2017.

[111] Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *STOC*, pages 253–264, New York, NY, USA, 1978. ACM.

[112] Junjie Ye. A note on finding dual feedback vertex set. *CoRR*, abs/1510.00773, 2015.

Homi Bhabha National Institute

Evaluation Report of Ph.D. Viva Voce

Board of Studies in Mathematical Sciences

1. Constituent Institution: **Institute of Mathematical Sciences, Chennai**

2. Name of the Student: Lawqueen Kanesh

3. Enrolment Number: MATH10201504008

4. Date of Enrolment in HBNI: Aug. 1, 2015

5. Date of Submission of Thesis: June. 06, 2020

6. Title of the Thesis: Parameterized Complexity of Conflict-Free Solutions,

7. Number of Doctoral Committee Meetings held with respective dates:
Number of Meetings held 3.
March 17th, 2017,
July 27th, 2018,
May 21st, 2019

8. Name and Affiliation of Examiner 1: Meirav Zehavi, Ben-Gurion University, Israel

Recommendation of the Examiner 1 (Thesis Evaluation) (i) accepted, (ii) accepted after revision, or (iii) rejected:

9. Name and Affiliation of Examiner 2: L. Sunilchandran, Indian Institute of Science, Bangalore

Recommendation of the Examiner 2 (Thesis Evaluation) (i) accepted, (ii) accepted after revision, or (iii) rejected:

Recommendations of the Viva Voce Board

1. Date of Viva Voce Examination: Dec. 01, 2020

2. Recommendations for the award of the Ph.D. degree: Recommended/ Non Recommended
(If recommended, give summary of main findings and overall quality of the thesis)

In this thesis the candidate studies algorithms mainly in the parameterized complexity framework. In her thesis she introduced a new variant for some of the classical problems in Graph Algorithms, which is called conflict-free version, and studied them from the viewpoint of classical and Parameterized Complexity. In the second part of the thesis she extended the Feedback Vertex Set (FVS) problem to Hypergraphs and studied from the view point of Parameterized Complexity. The thesis at hand is at the frontier of parameterized algorithms and complexity and contributes substantially in moving the boundaries forward.

The candidate was asked to give an overview of her research work and was then questioned on the same. The board of examiners also enquired about the various suggestions and corrections mentioned in the reports of the thesis referees and assured themselves that these were all addressed and that the necessary modifications were incorporated in the final copy of the thesis which was submitted at the time of the examination.

- In view of the good quality of the research work,
- in view of the satisfactory reports by the thesis referees (reports attached), and
- in view of the good performance of the candidate in the *viva voce* examination,

it is recommended that the thesis submitted by **Ms Lawqueen Kanesh** be **accepted** for the award of the degree of **Doctor of Philosophy (Ph.D.)** by the **Homi Bhabha National Institute (HBNI)**.

Names and signatures of the viva voce board:

S. No.	Designation	Name	Signature	Date
1.	Chair	Venkatesh Raman		Dec. 01, 2020
2.	Convener/Guide	Saket Saurabh		Dec. 01, 2020
3.	Member 1	R. Ramanujam		Dec. 01, 2020
4.	Member 2	Geevarghese Philip		Dec. 01, 2020
5.	Member 3	Vikram Sharma		Dec. 01, 2020
6.	External member	L. Sunilchandran		Dec. 01, 2020

Dean Academic, CI

Thesis Highlight

Name of the Student: Lawqueen Kanesh
Name of the CI/OCC: Dr. Saket Saurabh **Enrolment No.:** MATH10201504008
Thesis Title: Parameterized Complexity of Conflict-Free Solutions
Discipline: Mathematical Science **Subarea of Discipline:** Theoretical Computer Science
Date of viva voce: 01/12/2020

In this thesis we introduce and study *conflict-free* variants of several classical problems in Graph Algorithms and Parameterized Complexity and study them from the viewpoint of classical and Parameterized Complexity. We also extend the Feedback Vertex Set (FVS) problem to hypergraphs and study from the view point of Parameterized Complexity. We first study conflict-free versions of Vertex Deletion problems. Let Q be a family of graphs (or property) – such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs or planar graphs. In the Vertex Deletion problem corresponding to Q (Q-VD), given a graph G, and a non-negative integer k, the goal is to delete a set S of at most k vertices such that the remaining graph G-S after deletion of S belongs to the family of graphs Q. In the conflict-free version of the Vertex Deletion problem corresponding to Q (Q-CF-VD), we are given a conflict graph H together with the graph G and an integer k, and the goal is to find a set S which is a subset of vertices of the graph G of size at most k, such that S is a solution to (G,k) of Q-VD and S is an independent set in H. We study the complexity of CF-Q-VD based on the forbidden set characterization of the property Q. (1) *CF-VD is FPT with finite forbidden characterization, W[1]-hard otherwise.* For graph properties with finite forbidden characterization, we show that CF-Q-VD is FPT and admits a polynomial kernel. For graph properties without finite forbidden characterization, we show that if Q is characterized by a "well-behaved" infinite family of forbidden induced subgraphs, then CF-Q-VD is W[1]-hard. In particular, we show that the conflict-free version of FVS (CF-FVS) is W[1]-hard even when G is a disjoint union of cycles. A similar result holds for the conflict-free version of Odd Cycle Transversal (CF-OCT), Chordal Vertex Deletion (CF-CVD) and Interval Vertex Deletion (CF-IVD). (2) *CF-VD with infinite forbidden characterizations is FPT for restricted conflict graphs*. We also show that CF-FVS, CF-OCT, CF-CVD, and CF-IVD are FPT, when H belongs to the family of d-degenerate graphs or nowhere dense graphs. (3) *Dichotomy for CF-FVS*. We obtain a complete dichotomy result on the Parameterized Complexity of the problem H-CF-FVS, where the conflict-free graph H belongs to graph class H (for hereditary H), in terms of the Independent Set problem. We show that H-CF-FVS is in FPT if and only if H+Cluster Independent Set is in FPT, where H+Cluster Independent Set is the Independent Set problem on the edge union graph of a cluster graph and a graph in H. (4) *Kernelization complexity for CF-FVS and CF-OCT*. We obtain a polynomial kernel for CF-FVS and show that CF-OCT does not admit a polynomial kernel, when H belongs to the family of d-degenerate graphs. (5) *Parameterized Complexity of conflict-free versions of Maximum Matching and Shortest Path.* We also study conflict-free (parameterized) versions of the Maximum Matching (CF-MM) and Shortest Path (CF-SP) problems. We show that both CF-MM and CF-SP are W[1]-hard, when parameterized by the solution size. For the CF-MM problem, we give an FPT algorithm, when the conflict graph belongs to the family of chordal graphs. For conflict graphs being d-degenerate and nowhere dense graphs, we obtain FPT algorithms for both CF-MM and CF-SP. We study a variant of CF-MM and CF-SP, where instead of conflicting conditions being imposed by independent sets in a conflict graph, they are imposed by independence constraints in a (representable) matroid. We give FPT algorithms for the above variant of both CF-MM and CF-SP. (6) *FVS in hypergraphs*. Finally, we extend our study from problems on graphs to hypergraphs. In particular, we study the Feedback Vertex Set problem on hypergraphs. We show that FVS on hypergraphs is W[2]-hard, when parameterized by the solution size. We obtain FPT algorithms for FVS on hypergraphs, when the input hypergraph is restricted to d-hypergraphs and linear hypergraphs.