Parameterized approximation algorithms for Partial Covering and Satisfiability with budget constraints

By
Anannya Upasana

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

June 16, 2026

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 **Anannya Upasana** entitled "Parameterized approximation algorithms for
Partial Covering and Satisfiability with budget constraints" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

(VIKRAM SHARMA) Date: **June 16, 2026**
Chairman -

(SAKET SAURABH) Date: **June 16, 2026**
Guide/Convenor -

(ARIJIT GHOSH) Date: **June 16, 2026**
Examiner -

(SUSHMITA GUPTA) Date: **June 16, 2026**
Member 1 -

(C RAMYA) Date: **June 16, 2026**
Member 2 -

(PRANABENDU MISRA) Date: **June 16, 2026**
Member 3 -

Date: **June 16, 2026**
Member 4 -

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: June 16, 2026

Place: IMSc, Chennai

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.

Anannya Upasana

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.

Anannya Upasana

CERTIFICATION ON ACADEMIC INTEGRITY

Undertaking by the Student

1. I, Anannya Upasana,_____(Name of the student), HBNI Enrolment No. MATH10202104003_____hereby undertake that

_____

_____

the Thesis, titled " Parameterized approximation algorithms for Partial Covering and Satisfiability with budget constraints

" is prepared by me and is the original work undertaken by me.

_____

2. I also hereby undertake that this document has been duly checked through a plagiarism detection tool and the document is found to be plagiarism free as per the guidelines of the Institute/ UGC.

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

Anannya 16.06.2026

Signature of the Student with date

Endorsed by the Thesis Supervisor:

I certify that the thesis written by the researcher is plagiarism free as mentioned above by the student.

Signature of the Thesis Supervisor with Date and Name: Saket Saurabh, 16th June 2026.
Designation: Professor
Department/ Centre: Theoretical Computer Science
Name of the CI/ OCC: The Institute of Mathematical Sciences

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Publications:

- **Published**

1. Pallavi Jain, Lawqueen Kanesh, Fahad Panolan, Souvik Saha, Abhishek Sahu, Saket Saurabh, Anannya Upasana.

   Parameterized Approximation Schemes for Biclique-free Max k-weight SAT and Max Coverage

   Proceedings of the 16^{th} *ACM-SIAM Symposium on Discrete Algorithms (SODA* 2023*)*, and *ACM Transactions of Algorithms 22(1): 3:1-3:30* 2026.

2. Pallavi Jain, Lawqueen Kanesh, Fahad Panolan, Souvik Saha, Abhishek Sahu, Saket Saurabh, Anannya Upasana.

   Max-SAT with Cardinality Constraint Parameterized by the Number of Clauses

   Proceedings of the 16^{th} *Latin American Symposium: Theoretical Informatics (LATIN* 2024*)*, and *Theoretical Computer Science (TCS) 1056: 115540* 2025.

3. Tanmay Inamdar, Pallavi Jain, Daniel Lokshtanov, Abhishek Sahu, Saket Saurabh, Anannya Upasana.

   Satisfiability to Coverage in Presence of Fairness, Matroid, and Global Constraints

   Proceedings of the 51^{st} *International Colloquium on Automata, Languages, and Programming (ICALP* 2024*).*

- **In Preparation**

1. Tanmay Inamdar, Pallavi Jain, Madhumita Kundu, Abhishek Sahu, Souvik Saha, Saket Saurabh, Anannya Upasana.

   Multi-Criteria Lossy Kernels for Fair and Matroid-Constrained Satisfiability

Anannya Upasana

Homi Bhabha National Institute

Certificate from the Guide on Publications

1. **Name of the Student:** Anannya Upasana
2. **Name of the Constituent Institution:** The Institute of Mathematical Sciences
3. **Enrolment No. and Date:** MATH10202104003, 01/01/2021
4. **Title of the Thesis:** Parameterized approximation algorithms for Partial Covering and Satisfiability with budget constraints
5. **Board of Studies:** Mathematical Sciences

1. The following are the publications resulting from the work reported in the thesis:

S. No.	Journal Publication(s)	Student's Contribution
1	Parameterized Approximation Schemes for Biclique-free Max k-weight SAT and Max Coverage [TALG 2025]	The student made contribution in terms of ideas and writing for the Maximum Coverage algorithm, wrote both the conference version (SODA) and the journal version, and led the TALG 2025 submission.
2	Max SAT with cardinality constraint parameterized by the number of clauses [TCS 2025]	The student contributed the main ideas for the reduction rules crucial to the kernelization algorithm, wrote both the conference version (LATIN) and the journal version, and led the TCS 2025 submission.

2. It is further certified that the part of the work included in this thesis will not be included again in any other thesis.

3. It is also certified that the co-authors of the above papers have no objection for including the indicated parts of the work in the present thesis.

Signature of the Guide
Date: 19/09/2025

To
HBNI Office

List of presentations and participation at conferences

1. Presented **Parameterized Approximation Schemes for Biclique-free Max k-weight SAT and Max Coverage** at the SODA 2023 conference in Florence, Italy.

2. Presented a poster on **Parameterized Approximation Schemes for Biclique-free Max k-weight SAT and Max Coverage** at the ACM ARCS 2024 event at NISER Bhubaneswar, India.

Anannya Upasana

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Saket Saurabh, for his constant guidance, encouragement, and patience throughout my PhD. His insight, clarity of thought, and unwavering support have played a central role in shaping this thesis and my development as a researcher. Beyond specific research directions, he has taught us *how* to do research—how to read critically, comprehend the intuition behind ideas, write clearly, and collaborate effectively with others, among many things. His treatment of research as an art form, emphasizing both rigor and creativity has taught me a lot. He has always been a thoughtful listener, offering guidance that helped me identify gaps, build confidence, and steadily develop myself.

I am grateful to the faculty members at the Institute of Mathematical Sciences (IMSc) who taught and influenced me during my PhD years: Dr. Meena Mahajan, Dr. Venkatesh Raman, Dr. R. Ramanujam, Dr. Sushmita Gupta, Dr. Vikram Sharma, Dr. C. R. Subramanian, Dr. C. Ramya, Dr. Prakash Saivasan, and Dr. Akanksha Agrawal. I also sincerely thank the faculty at the Indian Statistical Institute (ISI) during my Master's studies—Dr. Arijit Bishnu, Dr. Sourav Chakraborty, Dr. Arijit Ghosh, Dr. Sushmita Sur-Kolay, Dr. Subhas Nandy and, Dr. Sandip Das—for providing a strong academic foundation through their courses and mentorship.

I would like to express my particular gratitude to Dr. Arijit Bishnu and Dr. Gopinath Mishra for taking me under their wing during my early years in research. Dr. Bishnu's patience, encouragement, and belief in me, along with the opportunity to work on my first research problem, played a crucial role in shaping my interest in research and setting me on the path that ultimately led to this thesis.

I would like to acknowledge my collaborators and co-authors for the stimulating discussions, shared ideas, and fruitful collaborations that significantly enriched my research. In particular, I thank Dr. Pallavi Jain, Dr. Lawqueen Kanesh, Dr. Roohani Sharma, Dr. Fahad Panolan, Dr. Abhishek Sahu, Dr. Tanmay Inam-

dar, Dr. Matthias Bentert, Dr. Satyabrata Jana, Dr. Nidhi Purohit, as well as Souvik Saha, Madhumita Kundu, Sobyasachi Chatterjee, Sanjay Seetharaman, and Subhada Aute. I am especially grateful to Souvik, Lawqueen, Abhishek, Tanmay, Roohani, Madhumita, Sobyasachi, and Sanjay for many fruitful discussions that helped shape my thinking and approach to several problems.

I am especially thankful to Dr. Fahad Panolan for hosting me at IIT Hyderabad during the summer of 2022, and to Dr. Fedor Fomin, Dr. Petr Golovach, and Dr. Saket Saurabh for hosting me at the University of Bergen, Norway, during the fall of 2024. These research visits provided valuable opportunities for collaboration and academic exchange.

I gratefully acknowledge the Google Travel Grant 2023 and the SODA 2023 Student Travel Grant for supporting my travel and stay in Florence, Italy, during the SODA 2023 conference. I also thank the administrative and support staff at IMSc for providing an excellent research environment and for ensuring access to facilities that made my stay comfortable and productive.

I would also like to shout out to Sonam, PC, Patro, and Souvik for being such wonderful friends. A special mention goes to PC for making my stay in Rome truly memorable—it felt as though no time had passed at all, despite us meeting again after so long. I am also grateful to Souvik for all the gastronomic and cycling adventures, which added an extra layer of joy to these memories.

Finally, I express my deepest gratitude to my family—my father, Mr. B. K. Mangaraj, my mother, Mrs. Binita Mangaraj, and my sister, Aishwarya Aradhana—for their unwavering love, patience, and encouragement throughout this journey. Their support has been a constant source of strength and motivation.

Contents

Abstract

Many fundamental optimization problems are NP-hard, making it unlikely that they admit algorithms that are both exact and efficient. Two such canonical problems are Maximum Coverage and Maximum Satisfiability with Cardinality Constraint (CC-MaxSat), which lie at the heart of partial covering. Although both problems admit polynomial-time approximation algorithms with approximation ratio $(1 - 1/e)$, which is optimal assuming P≠NP, this barrier persists even in the parameterized setting. This motivates the search for algorithms that exploit additional structural properties of the input to obtain significantly stronger guarantees.

This thesis studies parameterized approximation algorithms for partial covering problems through a unified framework centered around Maximum Coverage and CC-MaxSat. We investigate two complementary optimization objectives: maximizing the number of satisfied constraints using a bounded number of selected objects, and minimizing the number of selected objects required to satisfy a prescribed number of constraints. We further extend these problems by incorporating fairness constraints on the constraints to be satisfied and matroid constraints on the selected objects, yielding a broad family of generalized optimization problems.

For the first optimization objective, we develop Efficient Parameterized Approximation Schemes (EPAS) for CC-MaxSat and Maximum Coverage on structurally sparse instances such as $K_{d,d}$-free formulas and set systems respectively. A central conceptual contribution of the thesis is a randomized approximation-preserving reduction establishing the equivalence of CC-MaxSat and Maximum Coverage in the context of parameterized approximation. This equivalence extends naturally to generalized variants with fairness and matroid constraints, allowing algorithmic techniques developed for one problem to be transferred to the other. Building upon this framework, we obtain EPAS for fair, matroid-constrained, and combined fair-matroid variants, significantly extending the scope of parameterized approximation

beyond previously known settings.

We also investigate parameterization by the number of satisfied constraints, obtaining improved fixed-parameter algorithms for Decision CC-MaxSat together with polynomial kernels on structurally sparse $K_{d,d}$-free instances. For the complementary optimization objective, we design parameterized approximation schemes and establish matching hardness results. This extends to the more general variant with fairness constraints as well. Thus, we provide a nearly complete understanding of the approximability landscape under this objective.

Finally, we introduce the first framework for multi-criteria lossy kernelization, extending classical lossy kernelization to optimization problems with multiple simultaneous constraints. Using this framework, we obtain polynomial lossy kernels for generalized variants of CC-MaxSat with fairness and matroid constraints, resolving an open question regarding the existence of lossy kernels for these problems.

Overall, this thesis establishes new algorithmic connections between satisfiability and covering problems, develops a unified framework for parameterized approximation under increasingly expressive constraints, and advances the state of the art in parameterized approximation algorithms and kernelization for partial covering problems.

List of Figures

List of Tables

Part I: Introduction

Chapter 1

From Intractability to Parameterized Approximation

This thesis studies partial coverage problems through the lens of parameterized approximation. In this chapter, we introduce the algorithmic paradigms developed to cope with computational intractability and formally describe the problems that constitute the focus of this work.

1.1 Dealing with intractability

The modern notion of algorithmic efficiency was formalized independently by Cobham [2] and Edmonds [3] in the 1960s. An optimization problem is said to be efficiently solvable if there exists an algorithm that computes an optimal solution in time polynomial in the input size.

More precisely, an algorithm is considered *efficient* if:

- the value of the solution obtained by the algorithm is optimum,
- the time taken by the algorithm is a polynomial function of the input size, and
- the algorithm solves all instances of the problem.

However, the discovery of NP-Hard problems [4, 5, 6] demonstrated that many natural combinatorial problems are unlikely to admit algorithms satisfying all three criteria simultaneously, unless P = NP. This realization led to the development of

several complementary algorithmic frameworks, each relaxing a different notion of efficiency.

1.1.1 Approximation Algorithms

Approximation algorithms relax the requirement of optimality while retaining polynomial running time [7, 8].

For a minimization problem, an algorithm is an α-approximation if it returns a solution whose value is at most α times the optimum. For maximization problems, it must return a solution whose value is at least $1/\alpha$ times the optimum, where $\alpha > 1$.

The goal of approximation is to obtain provably near-optimal solutions efficiently. Over the years, this paradigm has led to polynomial-time approximation schemes (PTAS), constant-factor approximations, and tight inapproximability results that delineate the limits of polynomial-time computation.

1.1.2 Exact Exponential-Time Algorithms

Another way to cope with NP-hardness is to relax the requirement of polynomial running time while retaining optimality. Exact exponential-time algorithms aim to solve problems optimally while minimizing the exponential dependence on the input size [9].

This line of work refines brute-force search using techniques such as branching, measure-and-conquer analysis, and inclusion–exclusion methods. Although exponential in the worst case, these algorithms often significantly improve over naive enumeration.

1.1.3 Parameterized Complexity

Parameterized complexity relaxes the requirement that the algorithm must treat all parts of the input uniformly. Instead, it associates a parameter $k \in \mathbb{N}_{\geq 0}$ with each instance, where k typically captures structural or solution-related information and is expected to be small relative to the input size.

A problem is said to be fixed-parameter tractable (FPT) if it admits an algorithm

running in time $f(k)n^{\mathcal{O}(1)}$, where n is the input size and f is a computable function independent of n [10, 11].

Thus, any super-polynomial blowup is confined to the parameter k, isolating the combinatorial explosion to a controlled dimension of the input.

This framework has led to a refined complexity theory, including hardness classes such as W[1] and W[2], which provide evidence that certain parameterized problems are unlikely to be FPT.

1.1.4 Parameterized Approximation

While both approximation algorithms and parameterized complexity provide powerful tools, many problems resist both polynomial-time approximation and fixed-parameter tractability. For such problems, combining the two paradigms offers a natural way forward.

Parameterized approximation relaxes both optimality and efficiency simultaneously. A parameterized α-approximation algorithm computes an α-approximate solution in time $f(k)n^{\mathcal{O}(1)}$, where $\alpha > 1$ for a minimization problem and $\alpha < 1$ for maximization problem.

The goal is to exploit parameterized structure to surpass classical polynomial-time approximation guarantees.

To formalize this idea, we define two central notions:

A *Parameterized Approximation Scheme* (PAS) computes, for any $\varepsilon > 0$, a $(1-\varepsilon)$-approximate solution in time $f(k,\varepsilon)n^{g(\varepsilon)}$, where f and g are computable functions.

An *Efficient Parameterized Approximation Scheme* (EPAS) achieves the stronger running time $f(k,\varepsilon)n^{\mathcal{O}(1)}$.

The overarching objective of this framework is to go beyond what is achievable by polynomial-time approximation alone, by leveraging the additional structure captured by the parameter.

1.2 Introduction to partial coverage problems

We focus on the following two problems and their variants in this thesis.

- Max SAT with Cardinality Constraint (or CC-MaxSat), and
- Maximum Coverage.

Both problems fall under the broad umbrella of *partial covering*. Here, the input comprises of a collection of objects and a set of constraints to be met. Choosing an object satisfies a certain subset of constraints, and the goal is to select a specific number of objects, say k, so as to maximize the number of satisfied constraints.

CC-MaxSat is an important NP-Hard problem that has been studied across many algorithmic paradigms like approximation and parameterized complexity and generalizes well-known problems like Maximum Coverage, Partial Vertex Cover, Max 2-SAT with bisection constraint. We define the CC-MaxSat problem formally below.

Max SAT with Cardinality Constraint (or CC-MaxSat)
Input: A CNF formula $\Phi = (\mathcal{V}, \mathcal{C})$ on n variables and m clauses, and a positive integer k.
Parameter: k
Output: An assignment of weight at most k that satisfies the maximum number of clauses.

The weight of an assignment refers to the number of variables set to 1 by the assignment.

The second problem of interest is the Maximum Coverage problem, a monotone variant of CC-MaxSat where negative literals are not allowed. We define the Maximum Coverage problem formally below.

Maximum Coverage
Input: A set system $(\mathcal{U}, \mathcal{F})$ on n elements and m sets, and a positive integer k.
Parameter: k
Output: A subfamily of size at most k that covers the maximum number of elements.

We study two different optimization objectives for CC-MaxSat and Maximum Coverage:

- ***Objective* 1.** When the size of the solution, k, is fixed and the number of constraints that need to be satisfied is maximized.

- ***Objective* 2.** When the number of constraints that need to be satisfied, say t, is fixed and the number of objects we need to select to satisfy t constraints is minimized.

We call the version of CC-MaxSat with objective 2 as CC-MaxSat with Covering Constraint.

CC-MaxSat with Covering Constraint
Input: A CNF formula $\Phi = (\mathcal{V}, \mathcal{C})$ on n variables and m clauses, and a positive integer t.
Parameter: $\mathsf{OPT}_t(\Phi)$
Output: An assignment of minimum weight that satisfies at least t clauses.

Here, $\mathsf{OPT}_t(\Phi)$ is the minimum weight of any assignment that satisfies at least t clauses.

Similarly, we call the version of Maximum Coverage with objective 2 as Maximum Coverage with Covering Constraint.

Maximum Coverage with Covering Constraint
Input: A set system $(\mathcal{U}, \mathcal{F})$ on n elements and m sets, and a positive integer t.
Parameter: $\mathsf{OPT}_t(\mathcal{U}, \mathcal{F})$
Output: A subfamily of minimum size that covers at least t elements.

Here, $\mathsf{OPT}_t(\mathcal{U}, \mathcal{F})$ is the minimum sized subfamily that covers at least t elements.

Chapter 2

A brief survey of known results

This chapter presents a survey of the results that were known at the time the work in this thesis was initiated, thereby setting up the motivation for the work presented here.

Both CC-MaxSat and Maximum Coverage are well-known NP-Hard problems. There exists a greedy polynomial-time approximation algorithm with a factor of $(1 - \frac{1}{e})$[1] for Maximum Coverage which is optimal [12, 13]. That is, unless some well known complexity theory assumption fails, there is no polynomial time approximation algorithm for Maximum Coverage with ratio better than $(1 - \frac{1}{e})$. An approximation algorithm with the same factor also exists for CC-MaxSat running in polynomial time, which is optimal [14]. When the input formula is 2-CNF, the problem is called CC-Max-2-Sat. This problem has also been studied extensively [15, 16, 17, 14]. The best algorithm up to date for CC-Max-2-Sat achieves an approximation factor of roughly 0.929 [17], which improved over the series of results presented in [15, 16, 14]. Austin and Stanković [18] showed that CC-Max-2-Sat is hard to approximate within $0.929 + \varepsilon$, for any $\varepsilon > 0$, assuming Unique Games Conjecture (UGC). Further, a monotone variant (a version in which negated literals are not allowed) of CC-Max-2-Sat is also well studied with the name of Max k-Vertex Cover. Here, we are given a graph and the task is to select a subset of k vertices covering as many of the edges as possible. Note that Max k-Vertex Cover is a special case of Maximum Coverage where the frequency of each element is 2 and any pair of sets in the family have intersection at most 1. The problem is known to be approximable within 0.929 and is also hard to approximate within 0.929, assuming UGC [18, 19]. These results complete the picture from the

[1] Here, e is the base of the natural logarithm.

perspective of polynomial time approximation algorithms.

Thus, the natural next steps are to study the problem from the perspective of other algorithmic paradigms that are used to cope with NP-hardness, in particular from the perspective of Parameterized Complexity [11]. Both of these problems are W[2]-Hard parameterized by the solution size k [11]. That is, we do not expect CC-MaxSat (or Maximum Coverage) to admit an algorithm with running time $f(k)(n+m)^{\mathcal{O}(1)}$.

The above intractability results necessitate the study of these problems from a parameterized approximation viewpoint in the quest to improve the approximation factor obtained by the greedy polynomial-time algorithm [12]. Cohen-Addad et al. [20] initiated a study on this during their work on EPAS for k-Median and k-Mean and answered this in negative. Later, this was also studied by Manurangsi [21], who obtained the following strengthening over [20].

Proposition 1 (Hardness for Maximum Coverage [21])**.** *For any constant $\varepsilon > 0$ and any function h, assuming Gap-ETH, no $h(k)(n+m)^{o(k)}$ time algorithm can approximate* Maximum Coverage *with n elements and m sets to within a factor $(1 - \frac{1}{e} + \varepsilon)$, even with a promise that there exist k sets that fully cover the whole universe.*

Proposition 1 rules out any possibility of having an approximation algorithm for CC-MaxSat and Maximum Coverage with factor $(1 - \frac{1}{e} + \varepsilon)$ and running time $f(k, \varepsilon)(n+m)^{\mathcal{O}(1)}$ for any $\varepsilon > 0$. Thus, if we seek to design an approximation algorithm for CC-MaxSat and Maximum Coverage with factor $(1 - \frac{1}{e} + \varepsilon)$ and running in time $f(k, \varepsilon)(n+m)^{\mathcal{O}(1)}$, for any $\varepsilon > 0$, we must look for input set families that have some additional structure.

2.1 Exploiting structure: Results on structured instances

A natural and important special case of Maximum Coverage is the Max k-Vertex Cover problem. Given a graph G and an integer k, the objective is to select a set of k vertices that maximizes the number of edges incident to them. This can be viewed as a coverage instance where each edge is an element, and each vertex corresponds to the set of edges incident to it. In this formulation, every element appears in exactly two sets, and any two sets intersect in at most one element.

Marx [22] obtained the first EPAS for Max k-Vertex Cover parameterized by k, running in time $(k/\varepsilon)^{\mathcal{O}(k^3/\varepsilon)}(n+m)^{\mathcal{O}(1)}$. We now describe the fairly straightforward greedy procedure due to Marx.

The algorithm is based on a simple dichotomy depending on the maximum degree of the input graph. If the maximum degree is sufficiently large, then selecting the k highest-degree vertices already yields a $(1-\varepsilon)$-approximate solution. Intuitively, the sum of the degrees of these vertices is at least the sum of the degrees in an optimal solution, and although edges inside the chosen set may be double-counted (by at most $\binom{k}{2}$), this loss is negligible when the optimum is large. On the other hand, if the maximum degree is small, then the value of the optimum is also bounded. In this case, one can afford to search for the optimal solution exactly using color coding by trying all feasible values of the number of covered edges. Thus, the algorithm combines a simple greedy selection in the high-degree case with a bounded exhaustive search in the low-degree case, resulting in a clean and intuitive EPAS for Max k-Vertex Cover.

Later, Badanidiyuru, Kleinberg, and Lee [23] generalized this result to set systems with bounded VC-dimension, which is the largest class of structured instances for which Maximum Coverage is known to admit an EPAS parameterized by k. Their algorithm runs in time $2^{\mathcal{O}\left(\frac{k^2 d}{\varepsilon^5}\cdot(\ln(kd/\varepsilon))^{\mathcal{O}(1)}\right)}(n+m)^{\mathcal{O}(1)}$.

Another important special case of Maximum Coverage is when the input set family has bounded frequency. In this setting, we are given a set system $(\mathcal{U},\mathcal{F})$ such that each element of $\mathcal{U}$ appears in at most d sets, along with an integer k, and the objective is to select a k-sized subfamily that maximizes the number of covered elements. Skowron and Faliszewski [24] showed that this case admits an EPAS parameterized by k: for any $\varepsilon > 0$, there is an algorithm running in time $\left(\frac{d}{\varepsilon}\right)^{\mathcal{O}(k)}(n+m)^{\mathcal{O}(1)}$ that returns a subfamily of size k covering at least $(1-\varepsilon)\mathsf{OPT}_k(\mathcal{U},\mathcal{F})$ elements.

The algorithm is based on a simple structural observation. Let $S_1, S_2, \ldots, S_m$ be the sets of $\mathcal{F}$ ordered in non-increasing order of their cardinalities, and let $\mathcal{S}_{k'} = \{S_1, \ldots, S_{k'}\}$, where $k' = \frac{2kd}{\varepsilon} + k$ for $\varepsilon > 0$. The key insight is that an optimum solution can be gradually transformed into a solution entirely inside $\mathcal{S}_{k'}$ by replacing each set outside $\mathcal{S}_{k'}$ with one from $\mathcal{S}_{k'}$ so that the loss in coverage at each step is tightly bounded. Since each element appears in at most d sets, each replacement can only lose a bounded amount of covered elements, and in total the accumulated loss is at most an ε-fraction of the optimum. Consequently, there is always a k-sized subset of $\mathcal{S}_{k'}$ whose coverage is at least $(1-\varepsilon)$ fraction of the optimum, and it

suffices to enumerate all such subsets and return the best one. This yields a clean and intuitive EPAS for bounded-frequency Maximum Coverage.

Notice that in the case of d-set systems, i.e., set families where the set sizes are bounded by d, any k sets can cover at most kd elements. Bläser's result implies a $2^{O(kd)}(n+m)^{O(1)}$-time algorithm for Maximum Coverage on d-set systems [25]. Unlike the case for set systems with bounded frequency (when every element of the universe occurs in bounded number of sets), Maximum Coverage is FPT with respect to k on set systems with bounded set sizes.

Manurangsi [19] gave a very simple and clean EPAS for the weighted Max k-Vertex Cover problem with running time $(1/\varepsilon)^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, which was an improvement over Marx [22]. The main idea is structural and closely parallels the approach of Skowron and Faliszewski [24] for Maximum Coverage on bounded frequency set families. The algorithm restricts attention to a carefully chosen small candidate set consisting of the top $k + \lceil k/\varepsilon \rceil$ vertices ordered by non-increasing weighted degree. Intuitively, vertices with small weighted degree cannot significantly contribute to the coverage, and hence an almost-optimal solution can be found entirely among these high-degree vertices.

The argument proceeds by showing that any optimal solution can be transformed into one contained within this restricted set with only a small loss in total covered weight. This replacement argument is conceptually similar to the bounded-frequency analysis of Skowron and Faliszewski and a careful probabilistic analysis is used to show that if the candidate set is large enough, then randomly choosing replacement vertices bounds the cumulative loss in weighted degree. Consequently, it suffices to enumerate all k-subsets of this $O(k/\varepsilon)$-sized candidate set and return the best one, yielding a $(1-\varepsilon)$-approximation in time $(1/\varepsilon)^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

The EPAS by Skowron and Faliszewski [24] for set systems of bounded frequency as well as the EPAS for Max k-Vertex Cover by Manurangsi [26] have more efficient running times compared to that of Badanidiyuru, Kleinberg, and Lee [23]. This leads us to ask if there are other subclasses of instances with bounded VC-dimension where the running time of the EPAS can be made more efficient.

We seek inspiration from similar studies undertaken for Set Cover and Backdoor Sets [27, 28, 1]. It is well known that Set Cover is W[2]-Hard but admits an FPT algorithm when the incidence graph of the set system excludes some $K_{d,d}$ as a subgraph [28, 1]. Here, incidence graph is a bipartite graph, where one part, say A, has a vertex for each element in $\mathcal{F}$ and the other part, say B, has a vertex for

each element in U, and there is an edge between $a \in A$ and $u \in B$, if u belongs to the set corresponding to a. Further, $K_{d,d}$ denotes a complete bipartite graph with bipartitions of size d each.

This motivates us to study Maximum Coverage on $K_{d,d}$-free set systems and CC-MaxSat on $K_{d,d}$-free formulae. A formula Φ is said to be $K_{d,d}$-free if the *clause variable incidence graph* $G_\Phi = (A \cup B, E)$ excludes $K_{d,d}$ as a subgraph. A clause-variable incidence graph $G_\Phi = (A \cup B, E)$ associated with a formula Φ is a bipartite graph with a vertex in A for each variable and a vertex in B for each clause. If a clause c contains either a variable v or its negation $\neg v$, then there is an edge between $x_v \in A$, corresponding to v, and $y_c \in B$, corresponding to c.

An advantage for considering $K_{d,d}$-free instances is the following. $K_{d,d}$-free graphs (also called biclique-free graphs) is a very wide class of graphs and includes many well-studied graph classes such as *bounded treewidth graphs, graphs that exclude a fixed minor, graphs of bounded expansion, nowhere-dense graphs and graphs of bounded degeneracy*. That is, for any of the classes – bounded treewidth graphs, graphs that exclude a fixed minor, graphs of bounded expansion, nowhere-dense graphs and graphs of bounded degeneracy, there is a p such that the class is contained in the class of $K_{p,p}$-free graphs (see Figure 2.1 for an illustration of the inclusion relation between these classes). Set systems of bounded VC dimension form a more general class compared to set systems with bounded frequency, and even biclique-free set systems, which is the theme of the problems described in this work.

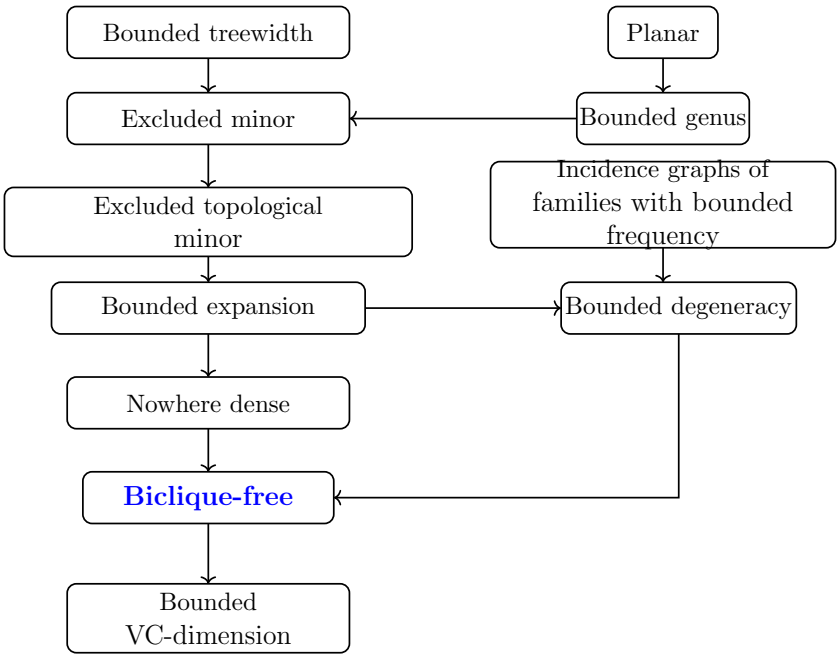

Figure 2.1: Inclusion relation between the class of biclique-free graphs considered in this paper and other well-studied graph classes (based on [1]). An arrow $A \to B$ indicates that class A is a subclass of B.

Chapter 3

Overview of results

This chapter presents a consolidated overview of the results in this thesis and also highlights how the different components fit together. The results are organized around two complementary optimization objectives: maximizing the number of satisfied constraints subject to a fixed solution size and minimizing the number of objects selected to satisfy a fixed number of constraints.

In the previous chapter, we motivated the study of EPAS for CC-MaxSat and Maximum Coverage on $K_{d,d}$-free instances parameterized by the solution size. We begin by presenting EPAS for both problems followed by a reduction that establishes their equivalence in the context of FPT approximation. We then discuss extensions under additional constraints, comprising of fairness constraints on the constraints to be satisfied and matroid constraints on the objects to be selected, and present the results for this generalization. We also consider an alternative parameterization by the number of satisfied constraints, obtaining both FPT algorithms on general instances and polynomial kernels on $K_{d,d}$-free instances.

Subsequently, we turn to the second optimization objective and present PAS and complement it with hardness result. Finally, we introduce a new multi-criteria lossy kernelization framework tailored to accommodate the constraints of our problem and give a polynomial lossy kernel for fair and matroid-constrained variant of CC-MaxSat in this new framework. The detailed proofs and technical developments are deferred to the subsequent chapters.

3.1 Objective 1: Optimizing the number of satisfied constraints

We begin by giving an EPAS for CC-MaxSat on $K_{d,d}$-free formulae running in time $2^{\mathcal{O}((\frac{dk}{\varepsilon})^d)}(n+m)^{\mathcal{O}(1)}$ and a more efficient EPAS for Maximum Coverage on $K_{d,d}$-free set systems running in time $\left(\frac{dk}{\varepsilon}\right)^{\mathcal{O}(dk)}(n+m)^{\mathcal{O}(1)}$. Our approach for the EPAS for CC-MaxSat on $K_{d,d}$-free set systems builds on the state-of-the-art EPAS for Max k-Vertex Cover and its generalizations [19, 24]. However, the broader generality of our setting, together with the fundamental differences between the monotone and non-monotone variants of the problem, necessitates substantial departures from their framework at several technically nontrivial points. While the EPAS for CC-MaxSat on $K_{d,d}$-free formulae also implies an EPAS for Maximum Coverage, using the fact that the latter does not have any negative literals, we derive a more efficient algorithm by reducing the problem to Max Red Blue Dominating Set.

Recasting to MaxRBDS and its advantages

We first recast Maximum Coverage to Max Red Blue Dominating Set (or MaxRBDS) with the help of incidence graph.

> Max Red Blue Dominating Set (MaxRBDS)
> **Input:** A bipartite graph G, with $V(G) = A \uplus B$, and an integer k.
> **Parameter:** k
> **Output:** A set $S \subseteq A$ of size at most k such that $|N(S)| = \mathsf{OPT}_k(G)$.

Here, $\mathsf{OPT}_k(G)$ denotes the maximum of $|N(S)|$, taken over k-sized subsets of A.

Observe that in the language of incidence bipartite graphs, finding a k-sized subfamily maximizing the number of covered elements corresponds to finding a k-sized subset $S \subseteq A$ such that $|N(S)|$ is maximized. Recasting the Maximum Coverage problem as MaxRBDS has another advantage. It also generalizes another well studied problem, namely, the hitting counterpart of Maximum Coverage– the Maximum Hitting Set problem.

MAXIMUM HITTING SET
Input: A family of sets, $\mathcal{F}$, over a universe $\mathcal{U}$, and a positive integer k
Parameter: k
Output: A subset $\mathcal{U}' \subseteq \mathcal{U}$ of size k such that the number of sets that intersects $\mathcal{U}'$ is maximized.

It is known that a decision variant of the MAXIMUM HITTING SET problem, DEC-MAXIMUM HITTING SET (where, additionally we are given an integer t, and we seek a set S of size k, that intersects at least t sets), is W[1]-Hard, even when each set in the family has size at most 2 [29]. When each set in the family has size at most 2, then MAXIMUM HITTING SET is popularly called PARTIAL VERTEX COVER. Marx gave the first EPAS for the PARTIAL VERTEX COVER problem which was followed by several algorithms with improved running time [22]. The current fastest algorithm was independently given by Manurangsi [19] and Skowron and Faliszewski [24], and runs in time $(\frac{1}{\varepsilon})^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. These algorithms can be generalised to set families where each set has size at most d.

Observe that in the language of incidence bipartite graph, finding a k-sized subset $\mathcal{U}' \subseteq \mathcal{U}$ such that the number of sets that intersects $\mathcal{U}'$ is maximized is same as finding a k-sized subset $B' \subseteq B$ such that $|N(B')|$ is maximized. Since the names "A" and "B" are interchangeable in the instance of MAXRBDS, an EPAS for this problem would imply an EPAS for both MAXIMUM COVERAGE and MAXIMUM HITTING SET. In particular, we give an EPAS for MAXRBDS running in time $\left(\frac{dk}{\varepsilon}\right)^{\mathcal{O}(dk)} n^{\mathcal{O}(1)}$.

To the best of our knowledge, the EPAS for MAXRBDS unifies and generalizes all known results regarding EPAS for either the MAXIMUM COVERAGE problem or the MAXIMUM HITTING SET problem, with the exception of the EPAS for set systems with bounded VC-dimension [23]. Note that there is an EPAS running in time $\left(\frac{1}{\varepsilon}\right)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ for MAX k-VERTEX COVER [19], a special case of MAXIMUM COVERAGE on $K_{2,2}$-free graphs, whereas our EPAS runs in $\left(\frac{k}{\varepsilon}\right)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ for MAX k-VERTEX COVER.

3.2 Equivalence of satisfiability and covering in the context of FPT Approximation

We next show that CC-MaxSat and Maximum Coverage are equivalent to each other in the context of EPAS parameterized by k when the objective is to maximize the number of satisfied constraints. In particular, we give a randomized reduction from CC-MaxSat to Maximum Coverage running in time $\mathcal{O}(1/\varepsilon)^k \cdot (n+m)^{\mathcal{O}(1)}$ that preserves the approximation guarantee up to a factor of $1-\varepsilon$, for any constant $\varepsilon \in (0,1)$. Furthermore, this reduction also works in the presence of fairness constraints on the satisfied clauses, as well as matroid constraints on the set of variables that are assigned true.

This allows us to focus on Maximum Coverage, rather than CC-MaxSat, at the expense of $\varepsilon^{-\mathcal{O}(k)}$ in the running time. Further, there is no assumption on the input formulas for this reduction to work. It immediately implies faster algorithms for CC-MaxSat by utilizing the known good algorithms for Maximum Coverage [19, 24, 30, 31].

By utilizing the EPAS for Maximum Coverage on $K_{d,d}$-free set systems described in Section 3.1, we obtain a randomized EPAS with running time $(\frac{dk}{\varepsilon})^{\mathcal{O}(dk)} n^{\mathcal{O}(1)}$ for CC-MaxSat on $K_{d,d}$-free formulae. Note that this EPAS is much faster, albeit randomized, than the one stated in Section 3.1. Furthermore, if the size of clauses is bounded by p, then CC-MaxSat admits a randomized EPAS with running time $(\frac{p}{\varepsilon})^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$. This result holds by utilizing the EPAS of Skowron and Faliszewski [24] for Maximum Coverage on set systems with bounded frequency. Both results hold with constant probability.

This reduction is robust enough that it can accommodate fairness constraints on the clauses or matroid constraints on variables. We explore these constraints in the next section. To be precise, one can give a similar reduction from $\mathcal{C}$-MaxSAT to $\mathcal{C}$-MaxCov, where $\mathcal{C} \in \{\text{M}, \text{F}, (\text{M}, \text{F})\}$ describes the constraint on the problem. Here, F stands for the fairness constraints on the clauses and M stands for the matroid constraints on variables. Note that in the presence of fairness constraints on the clauses, the success probability becomes $(\varepsilon/r)^{\mathcal{O}(k)}$, where r is the number of fairness constraints. Essentially, these reductions translate a constraint on the variables set to 1 (for CC-MaxSat and variants), to the corresponding family of sets (for Maximum Coverage and variants). Thus, at the expense of a multiplicative $(r/\varepsilon)^{\mathcal{O}(k)}$ factor in the running time, we can focus on Maximum Coverage and

its variants considered in the next section.

3.3 Optimizing the number of satisfied constraints subject to additional criteria

We generalize the Maximum Coverage problem in several directions by adding either fairness constraints on the elements covered or asking our solution to be an independent set of a given matroid.

3.3.1 Matroid Constraints

Note that Maximum Coverage is a special case of submodular function maximization subject to a cardinality constraint. In the latter problem, we are given (an oracle access to) a submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ [1], and the goal is to find a subset $U \subseteq V$ that maximizes $f(U)$ over all subsets of size at most k. Indeed, coverage functions are submodular and monotone (i.e., adding more sets cannot decrease the number of elements covered). There has been a plethora of work on monotone submodular maximization subject to cardinality constraints, starting from Wolsey [32]. In a further generalization, we are interested in monotone submodular maximization subject to a matroid constraint – in this setting, we are given a matroid $\mathcal{M} = (\mathcal{U}, I)$ [2] via an *independence oracle*, i.e., an algorithm that answers queries of the form "Is $P \in I$?" for any $P \subseteq \mathcal{U}$ in one step, and we want to find an independent set $S \in I$ that maximizes $f(S)$. Note here that a uniform matroid of rank k [3] exactly captures the cardinality constraint. Calinescu et al. [33] gave an optimal $(1 - 1/e)$-approximation.

More recently, Huang and Sellier [30] and Sellier [31] studied the problem of maximizing a coverage function subject to a matroid constraint, called Matroid Constrained Maximum Coverage. In this problem, which we call M-Maximum Coverage (M for "matroid" constraint), we are given a set system $(\mathcal{U}, \mathcal{F})$ and a matroid $\mathcal{M} = (\mathcal{F}, I)$ of rank k, and the goal is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ such

[1] $f : 2^V \to \mathbb{R}$ is submodular if it satisfies $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$

[2] Recall that a matroid is a pair $\mathcal{M} = (\mathcal{U}, I)$, where U is the ground set, and I is a family of subsets of $\mathcal{U}$ satisfying the following three axioms: (i) $\emptyset \in I$, (ii) If $A \in I$, then $B \in I$ for all subsets $B \subseteq A$, and (iii) for any $A, B \in I$ with $|B| > |A|$, then there exists an element $e \in B \setminus A$ such that $A \cup \{e\} \in I$. An element of I is an independent set in the matroid $\mathcal{M}$.

[3] Rank of a matroid is equal to the maximum size of any independent set in the matroid.

that $\mathcal{F}' \in I$ and $\mathcal{F}'$ maximizes the number of elements covered. Note that M-Maximum Coverage is a generalization of Maximum Coverage. In the latter paper, Sellier [31] designed an EPAS for M-Maximum Coverage, running in time $(d/\varepsilon)^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$ for frequency-d set systems. Note that this result generalizes that of [24, 19] from a uniform matroid consraint to an arbitrary matroid constraint of rank k.

Analogous to M-Maximum Coverage, one can define a matroid constrained version of CC-MaxSat, called M-Max k-weight SAT. In this problem, we are given a CNF-SAT formula Φ and a matroid $\mathcal{M}$ of rank k on the set of variables. The goal is to find an assignment that satisfies the maximum number of clauses, with the restriction that, the set of variables assigned 1 must be an independent set in $\mathcal{M}$. Note that M-Max k-weight SAT generalizes M-Maximum Coverage as well as CC-MaxSat.

3.3.2 Fairness or Multiple Coverage Constraints

We consider an orthogonal generalization of Maximum Coverage. Note that an optimal solution for Maximum Coverage may leave many elements uncovered. However, such a solution may be deemed *unfair* if the elements are divided into multiple colors (representing, say, people of different demographic groups), and the set uncovered elements are biased against a specific color. To address these constraints, the following generalization of Maximum Coverage, which we call F-Maximum Coverage (F stands for fair), has been studied in the literature. Here, we are given a set system $(\mathcal{U}, \mathcal{F})$, a coloring function $\chi : \mathcal{U} \to [r]$, a coverage requirement function $t : [r] \to \mathbb{N}$, and an integer k; and the goal is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ of size at most k such that, for each $i \in [r]$, the union of elements in $\mathcal{F}'$ is at least $t(i)$ (or t_i).

Since F-Maximum Coverage is a generalisation of Maximum Coverage, it inherits all the lower bounds known for Maximum Coverage. Furthermore, we can mimic the algorithm for Maximum Coverage parameterized by t (where you want to cover at least t elements with k sets) [25] to obtain an algorithm for F-Maximum Coverage parameterized by $\sum_{j\in[r]} t_j$. However, the problem is NP-Hard even when $t_j \leq 1$, $j \in [r]$, via a simple reduction from Set Cover.

F-Maximum Coverage has been studied under multiple names in the approximation algorithms literature; however much of the focus has been on approximating the

size of the solution, rather than the coverage. Notable exception include Chekuri et al. [34] who gave a bicriteria approximation, that outputs a solution of size at most $\mathcal{O}(\log r/\varepsilon)$ times the optimal size, and covers at least $(1 - 1/e - \varepsilon)$ fraction of the required coverage of each color. Very recently, Bandyapadhyay et al. [35] recently designed an EPAS for F-Maximum Coverage for the set systems of frequency 2, running in time $2^{\mathcal{O}\left(\frac{rk^2 \log k}{\varepsilon}\right)} \cdot (n+m)^{\mathcal{O}(1)}$.

We give a randomized EPAS for F-Maximum Coverage running in time

$$\left(dr\left(\frac{\log k}{\varepsilon}\right)^r\right)^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$$

on set systems with frequency bounded by d. Note that this result generalizes the result of [35] to frequency-d set systems, and in the case of $d = 2$, our running time is faster than that of [35], albeit randomized. We also give a deterministic EPAS for F-Maximum Coverage running in time $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} \cdot (n+m)^{\mathcal{O}(1)}$ on $K_{d,d}$-free set systems.

One can also define the *fair* version of CC-MaxSat in an analogous way, which we call F-Max k-weight SAT. In this problem, we are given a CNF-formula Φ, a coloring function $\chi : \mathcal{C}_\Phi \to [r]$, a coverage demand function $t : [r] \to \mathbb{N}$, and an integer k. The goal is to find an assignment of weight at most k that satisfies at least $t(j)$ (also denoted as t_j) clauses of each color $j \in [r]$. By combining the EPAS parameterized by k for F-Maximum Coverage on frequency-d set systems with the reduction algorithm, we get an EPAS for F-Max k-weight SAT on d-CNF-SAT formulae with a similar running time as that for F-Maximum Coverage on frequency-d set systems. Similarly, by combining the EPAS parameterized by k for F-Maximum Coverage on $K_{d,d}$-free set systems with the reduction algorithm, we get an EPAS for F-Max k-weight SAT on $K_{d,d}$-free formulae with a similar running time as that for F-Maximum Coverage on $K_{d,d}$-free set systems.

3.3.3 Combining Matroid and Fairness Constraints

As discussed above, Maximum Coverage has been generalized in two orthogonal directions, namely, matroid constraints on the sets chosen in the solution, and fairness constraints on the elements covered by the solution. Although the corresponding variants of CC-MaxSat have not been studied in the literature, we mentioned that our techniques readily imply an EPAS for these problems for many

formulae with sparse structure. Given this, it is natural to ask if we can find good approximations for the variants of CC-MaxSat (resp. Maximum Coverage) that combines the two orthogonal generalizations, namely, matroid constraint on the variables assigned 1, and fairness constraints on the satisfied clauses (resp. matroid constraint on the sets chosen in the solution, and fairness constraints on the elements covered).

In the following, we formally define the common generalization of M-Max k-weight SAT and F-Max k-weight SAT, which we call (M, F)-Max k-weight SAT.

(M, F)-Max k-weight SAT
Input. A CNF-SAT formula $\Phi = (\mathcal{V}, \mathcal{C})$ on n variables and m clauses, where the clauses $\mathcal{C}_\Phi$ of Φ are partitioned into r colors. Each color $j \in [r]$ has an associated demand t_j. Additionally, we are provided the independence oracle to a matroid $\mathcal{M} = (\mathcal{V}_\Phi, I)$ of rank k.
Parameter. k
Output. An assignment $\Psi : \mathcal{V}_\Phi \to \{0, 1\}$, such that $\Psi^{-1}(1) \in I$ and the number of clauses of color j satisfied by Ψ is at least t_j, for each $j \in [r]$.

In the special case where the CNF-SAT formula is monotone (i.e., does not contain negated literals), we obtain (M, F)-Maximum Coverage, which generalizes all the variants of Maximum Coverage discussed earlier.

(M, F)-Maximum Coverage
Input. A set system $(\mathcal{U}, \mathcal{F})$ on n elements and m sets, where the elements $\mathcal{U}$ are partitioned into r colors. Each color $j \in [r]$ has an associated demand t_j. Additionally, we are provided the independence oracle to a matroid $\mathcal{M} = (\mathcal{F}, I)$ of rank k.
Parameter. k
Output. A subfamily $\mathcal{F}' \subseteq \mathcal{F}$, such that $\mathcal{F}' \in I$ and the number of elements of color j covered by $\mathcal{F}'$ is at least t_j, for each $j \in [r]$.

We obtain a randomized EPAS for (M, F)-Maximum Coverage on set systems with maximum frequency d, that runs in time $\left(kdr\left(\frac{\log k}{\varepsilon}\right)^r\right)^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$ and returns a $(1-\varepsilon)$-approximation with at least a constant probability. We also obtain an EPAS for (M, F)-Maximum Coverage on $K_{d,d}$-free set systems, that runs in time $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} \cdot (n+m)^{\mathcal{O}(1)}$.

Finally, by reducing (M, F)-Max k-weight SAT on d-CNF formulas to (M, F)-Maximum Coverage with frequency d set systems, using the randomized reduction, and then using the results for (M, F)-Maximum Coverage, we obtain a randomized EPAS for (M, F)-Max k-weight SAT on d-CNF formulas, that runs in time $\left(kdr\left(\frac{\log k}{\varepsilon}\right)^r\right)^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$ and returns a $(1-\varepsilon)$-approximation with at least a constant probability. Analogously, by reducing (M, F)-Max k-weight SAT on $K_{d,d}$-free formulas to (M, F)-Maximum Coverage on $K_{d,d}$-free set systems, using the randomized reduction, and then using the results for (M, F)-Maximum Coverage, we obtain our most general result, which is a randomized EPAS for (M, F)-Max k-weight SAT on $K_{d,d}$-free CNF formulas, that runs in time $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} \cdot (n+m)^{\mathcal{O}(1)}$ and returns a $(1-\varepsilon)$-approximation with at least a constant probability.

Handling Multiple Matroid Constraints. Our results on M-Maximum Coverage and (M, F)-Maximum Coverage can be generalized to handle multiple matroid constraints on the solution, in the case when the matroids are *linear* or representable[4]. In this more general problem, we are given q linear matroids $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_q$, where $\mathcal{M}_i = (\mathcal{F}, I_i)$, each of rank at most k, and the solution S is required to be independent in all q matroids, i.e., $S \in \bigcap_{i\in[q]} I_i$. In this case, we can use linear algebraic tools ([36, 37]) to compute a representative set of size qk instead of k, and the computation requires $2^{\mathcal{O}(qk)} \cdot n^{\mathcal{O}(1)}$ time. Thus, EPAS for this problem now have a factor of q in the exponent.

Note that our EPAS improves upon the polynomial-time approximation guarantee of $1-1/e$ of Calinescu et al. [33] for monotone submodular maximization subject to a matroid constraint, in the special case of $K_{d,d}$-free coverage functions. To the best of our knowledge, this is the largest class of monotone submodular functions and matroid constraints for which the lower bound of $1-1/e$ can be overcome, even in FPT time. Further, the analogous results to (M, F)-Max k-weight SAT generalize these results to maximization of non-monotone/non-submodular functions.

We give a summary of how the various problems are related to each other, and a comparison of our results with the literature in Figure 3.1.

[4]A matroid $\mathcal{M} = (E, \mathcal{I})$ is representable over a field $\mathbb{F}$ if there exists a matrix M such that there exists a bijection between E and the columns on M with the property that, a subset $E' \subseteq E$ is independent in $\mathcal{M}$ iff the corresponding set of columns are linearly independent over $\mathbb{F}$.

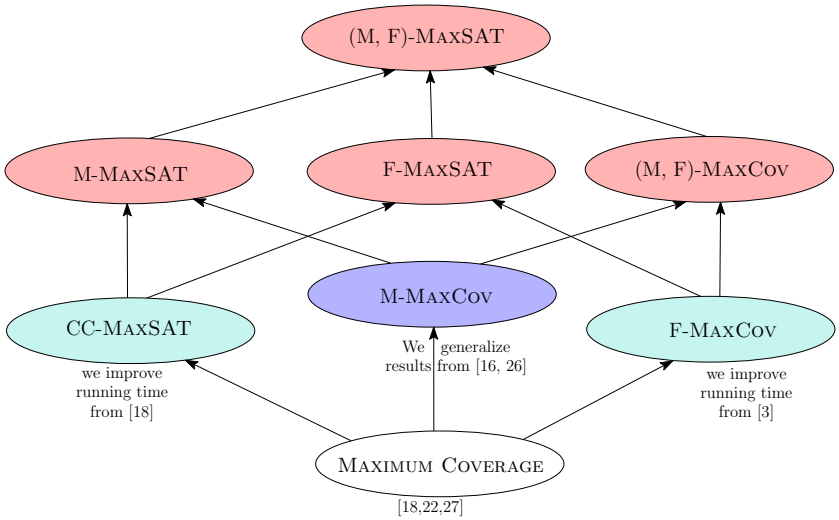

Figure 3.1: If there is an arrow of the form $A \to B$, then problem B generalizes problem A. EPAS for the problems in red bubbles are not known in the literature, and we study in this paper. For all the other problems EPAS are known in the literature for some cases. This paper improves the results in cyan and blue.

	frequency-d	**$K_{d,d}$-free**
Vanilla	-	$\left(\frac{dk}{\varepsilon}\right)^{\mathcal{O}(dk)} \cdot (n+m)^{\mathcal{O}(1)}$
Fair	$(dr)^{\mathcal{O}(k)} \cdot \left(\frac{\log k}{\varepsilon}\right)^{\mathcal{O}(kr)} \cdot (n+m)^{\mathcal{O}(1)}$	$2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})} (\log n)^{\mathcal{O}(rk)} \cdot (n+m)^{\mathcal{O}(1)}$
Fair + Matroid	$(kdr)^{\mathcal{O}(k)} \cdot \left(\frac{\log k}{\varepsilon}\right)^{\mathcal{O}(kr)} \cdot (n+m)^{\mathcal{O}(1)}$	$2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})} (\log n)^{\mathcal{O}(rk)} \cdot (n+m)^{\mathcal{O}(1)}$

Table 3.1: Summary of results for variants of Maximum Coverage (under Objective 1) for frequency-d and $K_{d,d}$-free set systems.

3.4 A different parameter: the number of satisfied constraints

We next study the decision version of both of our problems CC-MaxSat and Maximum Coverage with a different parameterization: the number of satisfied constraints t.

3.4.1 An FPT algorithm for Decision CC-MaxSat parameterized by the number of satisfied constraints

We define the Decision CC-MaxSat problem parameterized by t as follows.

Decision CC-MaxSat
Input: A CNF formula $\Phi = (\mathcal{V}, \mathcal{C})$ on n variables and m clauses, and two positive integers k and t.
Parameter: t
Output: An assignment of weight at most k that satisfies at least t clauses.

Bläser gave a $2^{O(t)}(n+m)^{\mathcal{O}(1)}$ algorithm for Decision Maximum Coverage, the decision version of Maximum Coverage, where given a set system $(U, \mathcal{F})$ and two positive integers k and t, the objective is to find k sets that cover at least t elements [25]. Bonnet et al. generalize this result to Decision CC-MaxSat by giving an FPT algorithm running in time $2^{O(t \log t)}(n+m)^{\mathcal{O}(1)}$ [38]. We give a different FPT algorithm for Decision CC-MaxSat with running time $2^{O(t)}(n+m)^{\mathcal{O}(1)}$. Our algorithm is faster than that of Bonnet et al.[38] and works on the following idea. The algorithm first generates a random assignment by independently setting each variable to a value in $\{0, 1\}$ uniformly at random. It then removes all clauses that are negatively satisfied by this assignment. A clause is said to be negatively satisfied if it contains a negative literal whose corresponding variable is assigned the value 0. Next, the remaining formula is transformed into an equivalent Maximum Hitting Set instance, and the solution to this instance is returned.

This procedure runs in polynomial time and succeeds with probability that is FPT in t. By standard probability boosting, this yields an FPT algorithm parameterized by t with constant success probability. Furthermore, the algorithm can be derandomized using (n, ℓ)-universal sets without increasing the asymptotic running time.

3.4.2 Polynomial kernels for $K_{d,d}$-free instances parameterized by the number of satisfied constraints

It is natural to investigate the existence of a polynomial kernel for Decision CC-MaxSat parameterized by t, following the result of an FPT algorithm for the problem with respect to t. Unfortunately, its monotone version, Decision Maximum Coverage, does not admit a polynomial kernel with respect to t unless $\mathsf{PH} = \Sigma_{\mathsf{p}}^{3}$ [39]. Thus, we cannot hope for a polynomial kernel with respect to t for Decision CC-MaxSat either. Again, this leads us to narrow our focus on input instances with additional structure that may admit a polynomial kernel parameterized by t.

Agrawal et al. [40] designed a polynomial kernel for a special case of Decision

Maximum Coverage when every element of the universe appears in at most d sets. In this kernel, the universe is bounded by $\mathcal{O}(dt^2)$, and the size of the family is bounded by $\mathcal{O}(dt)$. We generalize this result to $K_{d,d}$-free set systems by first recasting Decision Maximum Coverage to Decision Max Red Blue Dominating Set (or Dec MaxRBDS), the decision version of MaxRBDS, and then constructing a polynomial kernel of size $O(dt^{d+1})$ for the latter problem. Consequently, the existence of a polynomial kernel for Dec MaxRBDS on $K_{d,d}$-free bipartite graphs yields a polynomial kernel for Decision Maximum Coverage on $K_{d,d}$-free set systems.

Decision Max Red Blue Dominating Set
Input: A bipartite graph G, with $V(G) = A \uplus B$, and positive integers k, t.
Parameter: t
Output: A set $S \subseteq A$ of size at most k such that $|N(S)| \geq t$.

Furthermore, this kernelization for Dec MaxRBDS on $K_{d,d}$-free bipartite graphs is leveraged to improve the running time of the EPAS for MaxRBDS on the same graph class given in Section 3.1. Finally, to complete the picture with respect to this parameter, we extend these ideas to Decision CC-MaxSat on $K_{d,d}$-free formulae and obtain a kernel of size $\mathcal{O}(d4^{d^2}t^{d+1})$.

Considering other parameters. One may consider designing a polynomial kernel for Decision CC-MaxSat on $K_{d,d}$-free formulae with parameter k or $k+d$. Here, we would like to point out that Decision CC-MaxSat is W[1]-Hard when parameterized by $k+d$. This can be shown by a reduction from Partial Vertex Cover, which is W[1]-Hard parameterized by k [29]. The reduction is as follows [29]. In Partial Vertex Cover, we want to cover the maximum number of edges in the input graph using k vertices. The construction of a formula for Decision CC-MaxSat is as follows. We will have a variable x_v for each vertex v in the input graph G. For an edge $e = \{u, v\}$, we will construct a clause $C_e = (x_v \vee x_u)$. The resulting formula will be $K_{2,2}$-free for a simple graph G. This implies that the Decision CC-MaxSat problem is W[1]-Hard when parameterized by $k+d$.

3.5 Objective 2: Optimizing the solution size

Until now, we fixed the size of the solution and approximated the number of constraints that are covered, i.e., how many clauses we can satisfy when the weight of an assignment is fixed to be k or how many elements of the universe we can cover, if the number of sets that we are allowed to select is bounded by k. In literature, there is another version of these problems that has been studied, where the number of constraints that we need to cover is fixed but the number of objects that we can select to cover these constraints is optimized. For example, in the Set Cover problem, we have to cover all the elements of the universe and we optimize the number of sets that we select so as to cover the whole universe. For clarity of presentation, we call this version of CC-MaxSat and Maximum Coverage as CC-MaxSat with Covering Constraint and Maximum Coverage with Covering Constraint, respectively. For both problems, we *use k as a parameter*. These problems have been defined formally in Section 1.2.

Maximum Coverage with Covering Constraint, together with its special case on graphs called Partial Vertex Cover (where one wants to cover at least t edges with as few vertices as possible), has been studied extensively from the approximation front under the name of "partial covering" [41, 42, 43, 44]. It is known that the Partial Vertex Cover admits a 2-approximation, which cannot be improved further assuming UGC [41, 42, 43, 44, 45]. Partial Vertex Cover is also known to be W[1]-Hard parameterized by k, that is, it is unlikely to admit an FPT algorithm with respect to k [29]. Furthermore, both CC-MaxSat with Covering Constraint and Maximum Coverage with Covering Constraint generalize the classical Set Cover problem and hence we are unlikely to design any EPAS for these problems with respect to k [46]. This leads us to think about the families of instances on which CC-MaxSat with Covering Constraint or Maximum Coverage with Covering Constraint admit a PAS or an EPAS.

We solve the Maximum Coverage with Covering Constraint problem by recasting it to the Max Red Blue Dominating Set with Covering Constraint (or Max RBDS CovC) problem.

Max Red Blue Dominating Set with Covering Constraint
Input: A bipartite graph G, with $V(G) = A \uplus B$, and a positive integer t.
Parameter: $k = \mathsf{OPT}_t(G)$
Output: A set $S \subseteq A$ of size k, such that $|N(S)| \geq t$.

Here, $\mathsf{OPT}_t(G)$ denotes the minimum value of $|S|$ such that $|N(S)| \geq t$.

We show that Max RBDS CovC does not admit an EPAS parameterized by k and ε. Suppose, for the sake of contradiction, that such an EPAS exists. Then, for every $\varepsilon > 0$, there is an algorithm running in time $f(k, \varepsilon)n^{\mathcal{O}(1)}$ that produces a solution of size at most $(1 + \varepsilon)k$. Choosing $\varepsilon < \frac{1}{k}$ yields a solution of size strictly smaller than $k + 1$, and hence of size at most k, in $f(k)n^{\mathcal{O}(1)}$ time. Consequently, this would allow us to compute an optimal solution in FPT time, contradicting known complexity assumptions. Moreover, the problem Partial Vertex Cover, or equivalently Max RBDS CovC on $K_{2,2}$-free bipartite graph, does not admit an EPAS parameterized by k and ε by similar reasoning. This rules out the possibility of an EPAS parameterized by k and ε for Max RBDS CovC on $K_{d,d}$-free instances.

For Max RBDS CovC, we obtain a PAS on $K_{d,d}$-free bipartite graphs running in time $(kd)^{\mathcal{O}(kd)}n^{\mathcal{O}(1)}$. When $\varepsilon \geq \frac{1}{k}$, it gives a one-additive approximation algorithm and when $\varepsilon < \frac{1}{k}$, we can solve optimally by trying all possible sets of size at most k in $n^{\mathcal{O}(\frac{1}{\varepsilon})}$ time which is allowed in a PAS.

A PAS for Max RBDS CovC would imply a PAS for both Maximum Coverage with Covering Constraint and Maximum Hitting Set with Covering Constraint. For Maximum Coverage with Covering Constraint, given k and t, the algorithm either certifies that no subfamily of size at most k covers at least t elements, or returns a subfamily of size at most $k + 1$ that covers at least t elements. Similarly, for Maximum Hitting Set with Covering Constraint, the algorithm either guarantees that there is no hitting set of size at most k that hits at least t sets, or returns a hitting set of size at most $k + 1$ hitting at least t sets.

We generalize this PAS to incorporate fairness constraints as well. We denote Max RBDS CovC with fairness constraints as Partition Color Constrained Dominating Set (or PCCDS).

Partition Color Constrained Dominating Set (PCCDS)
Input: An instance $\mathcal{I} = (G, r, f, k, t)$, where $G = (A \uplus B, E)$ is a bipartite graph, $f : B \to [r]$ is a surjective function, a non-negative integer k, and for each color $j \in [r]$, a *coverage requirement* $t_j \geq 0$.
Here, $B_j \coloneqq \{\mathfrak{s} \in B : f(\mathfrak{s}) = j\}$ is a set of vertices of *color* j, for each $j \in [r]$.
Parameter: k
Question: Does there exist a subset $S \subseteq A$, such that (1) $|S| \leq k$, and (2) for each $j \in [r]$, $|N_j(S)| \geq t(j)$? Here, $N_j(S) \subseteq B_j$ is the set of vertices of color $j \in [r]$ that are adjacent to at least one vertex in S.

The PAS for PCCDS runs in time $(2^{\mathcal{O}(k^5 r^2 d \log r)} + (kd)^{\mathcal{O}(kd)})n^{\mathcal{O}(1)}$ and, either returns a set of size at $k + r$ that has at least t_j neighbors in B_j, for each $j \in [r]$, or it returns that there is no set of size at most k that has at least t_j neighbors in B_j, for each $j \in [r]$. We also show that this result cannot be improved to $(k + r - 1)$, unless FPT = W[1].

A PAS for PCCDS would imply a PAS for both F-Maximum Coverage and F-Maximum Hitting Set.

In contrast, no such algorithm exists under matroid constraints. Unless FPT = W[1], the guarantee is unattainable even for linear matroids and for set systems with frequency at most 2 and pairwise intersection at most 1. This is achieved via a reduction from Partial Vertex Cover with a matroid constraint on the vertex set.

	Vanilla	Fair	Matroid
$K_{d,d}$-free	$(kd)^{\mathcal{O}(kd)} \cdot (n+m)^{\mathcal{O}(1)}$	$\left(2^{\mathcal{O}(k^5 r^2 d \log r)} + (kd)^{\mathcal{O}(kd)}\right) \cdot (n+m)^{\mathcal{O}(1)}$	No PAS

Table 3.2: Summary of results for variants of Maximum Coverage with Covering Constraint (under Objective 2) for $K_{d,d}$-free set system.

3.6 Multi-criteria Lossy Kernelization

While the field of FPT approximation has been thriving over the past decade [21, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 26, 24, 59, 60, 61], the same cannot be said for lossy kernelization, its polynomial-time preprocessing counterpart.

A natural companion to FPT algorithms is polynomial-time preprocessing, also

known as *kernelization*. A parameterized problem is said to admit a *polynomial kernel* if there exists a polynomial-time algorithm (where the degree of the polynomial is independent of the parameter k), called a *kernelization algorithm*, that transforms any input instance into an equivalent instance whose size is bounded by a polynomial $p(k)$ in k, while preserving the answer. The resulting reduced instance is referred to as a $p(k)$-*kernel* for the problem. Once an FPT algorithm is established for a problem, a natural follow-up question is whether the problem also admits a polynomial kernel. The study of FPT algorithms and kernelization (also known as lossless preprocessing) has evolved in parallel, with significant synergies between the two. This has culminated in a comprehensive textbook on kernelization [11].

Analogous to kernelization in the exact setting, a companion notion of polynomial-time preprocessing for FPT-approximation is captured by the concept of *lossy kernels*, introduced by Lokshtanov et al. [62]. Loosely speaking, an (α)-approximate kernel of size $g(k)$ is a polynomial-time algorithm that, given an instance (I, k), outputs a new instance (I', k') such that $|I'|+k' \leq g(k)$, and any c-approximate solution s' to (I', k') can be efficiently transformed into a $(c \cdot \alpha)$-approximate solution s for the original instance (I, k). While fixed-parameter tractable (FPT) approximation algorithms have witnessed significant progress, their polynomial-time preprocessing counterpart—*lossy kernelization*—remains relatively underexplored, in contrast to the well-developed parallel between FPT algorithms and (exact) kernelization.

The next step, thus, seems to be to contribute to the development of lossy kernelization, thereby addressing this imbalance. We do so by studying lossy kernels for generalizations of Maximum Coverage and CC-MaxSat.

Following the EPAS parameterized by k for CC-MaxSat on biclique-free formulae, which extended prior work by Manurangsi [21] and Skowron and Faliszewski [24], a natural open question on the existence of a lossy kernel for the problem remained unsolved. Manurangsi [58] answered this question in the affirmative by designing a lossy kernel for CC-MaxSat on biclique-free instances. This lossy kernel encompasses the work of Sellier [31] who provided an EPAS and a lossy kernel, parameterized by k, for Maximum Coverage under bounded frequency and matroid constraints. Manurangsi [58] also posed the question of whether such a lossy kernel exists for a generalization of the problem that extends the cardinality constraint to more expressive settings, such as matroid constraints. This open question forms the starting point of our research on lossy kernels. Furthermore, Max k-Vertex Cover is also known to have lossy kernels [22, 62, 21].

We study lossy kernels for F-Max k-weight SAT and (M, F)-Max k-weight SAT. Since these problems are generalizations of F-Maximum Coverage and (M, F)-Maximum Coverage, we focus only on the variants of CC-MaxSat here. In particular, we study the following problem:

(M, F)-Max k-weight SAT
Input: An instance $\mathcal{J} = (\Phi, r, f, \mathcal{M}, k, t)$, where $\Phi = (\mathcal{V}, \mathcal{C})$ is a CNF-SAT formula, $f : \mathcal{C} \to [r]$ is a surjective function, and for each $j \in [r]$, we say that $\mathcal{C}_j := \{\mathfrak{s} \in \mathcal{C} : f(\mathfrak{s}) = j\}$ is the set of clauses of *color* j, and we have a *coverage requirement* $t_j := t(j) \geq 0$. Moreover, we have a matroid $\mathcal{M} = (\mathcal{V}, I)$ of rank k defined on the set of variables $\mathcal{V}$ where k is a non-negative integer.
Parameter: k
Output: An assignment β of weight at most k to $\mathcal{V}$, such that $\beta^{-1}(1) \in I$, and for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(\beta)| \geq t_j$.
Here, $\mathsf{sat}_{\Phi_j}(\beta) \subseteq \mathcal{C}_j$ is the set of clauses of color $j \in [r]$ that are satisfied by β in the subformula $\Phi_j = (\mathcal{V}, \mathcal{C}_j)$.

3.6.1 Multi-criteria Optimization Framework for Lossy Kernels

Clearly, in the (M, F)-Max k-weight SAT problem, there is more than one objective function: covering t_j clauses in each $\mathcal{C}_j$. The traditional lossy kernelization framework by Lokshtanov et al. is defined only for single criteria optimization function [62]. Thus, in order to obtain a lossy kernel for (M, F)-Max k-weight SAT, we first extend the framework by Lokshtanov et al. [62] to include more than one objective function.

Thus our contribution is conceptual: we extend the framework of lossy kernels to encompass multi-criteria optimization problems, following the framework initiated by Ravi et al. [63] and Grandoni et al. [64]. Their work introduced a model for multi-objective approximation that captures trade-offs across competing goals. Our generalized framework enables lossy kernelization for problems with multiple intertwined structural constraints—a setting frequently encountered in computational social choice theory, such as committee selection, fair division, or diversified ranking.

In particular, we give a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-approximate kernel for (M, F)-Max

k-WEIGHT SAT, with $\mathcal{O}\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(rd^2)}$ variables and $\mathcal{O}\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(rd^3)}$ clauses, when the formula is $K_{d,d}$-free. Here, $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$ is a tuple with $r+2$ elements. A $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-approximate kernel is a pair of algorithms — called the *reduction algorithm* and the *solution lifting algorithm* — defined for any $\varepsilon \in (0, 1)$.

Informally, a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-approximate solution refers to a set S of variables of minimum size (capturing the 1-factor) that satisfies at least $\frac{t_j}{1+\varepsilon}$ clauses of color j, for every $j \in [r]$, and additionally forms an independent set in the matroid defined on the variables.

The *reduction algorithm* takes an instance of (M, F)-MAX k-WEIGHT SAT and returns a reduced instance of size bounded by a polynomial function of the parameter k, running in time polynomial in the input size. The *solution lifting algorithm* takes as input the original instance, the reduced instance produced by the reduction algorithm, and a $\{\beta_0, \beta_1, \ldots, \beta_{r+1}\}$-approximate solution to the reduced instance for any $\beta_i \geq 1$, $i \in [r+1]$, and returns a $\{1 \cdot \beta_0, (1+\varepsilon) \cdot \beta_1, \ldots, (1+\varepsilon) \cdot \beta_r, 1 \cdot \beta_{r+1}\}$-approximate solution to the original instance.

This result generalizes the lossy kernels of Manurangsi [58] and Sellier [31].

3.7 Organization of the thesis

This thesis is organized into four parts, each addressing a different aspect of partial constraint satisfaction and covering problems.

Part I provides a high-level overview of the results developed in the thesis. Part II studies the problems CC-MAXSAT and MAXIMUM COVERAGE under the first objective, where the goal is to maximize the number of satisfied constraints while fixing the solution size. Part III considers the same problems under the second objective, which seeks to minimize the number of selected objects subject to satisfying a fixed number of constraints. Finally, Part IV introduces and discusses the multi-criteria lossy kernelization framework.

Before presenting these results, we introduce the necessary background. The fundamentals of parameterized complexity, along with results on combinatorial objects that are used throughout the thesis, are presented in Chapter 4.1. The notation and preliminaries required for the subsequent chapters appear in Chapter 4.3. We conclude the thesis in Part V, Chapter 12.

Part II begins with efficient parameterized approximation schemes (EPAS) for CC-MaxSat and Maximum Coverage on $K_{d,d}$-free instances, parameterized by the solution size k. The full details appear in Chapters 5.1 and 5.2, respectively. This is followed by a randomized reduction from CC-MaxSat to Maximum Coverage, also parameterized by k, the details of which are presented in Chapter 6.

We then study variants of these problems under additional constraints, namely fairness and matroid constraints. Since the corresponding results for CC-MaxSat can be obtained using the reduction from Chapter 6, we focus primarily on variants of Maximum Coverage. The main algorithmic ideas underlying these extensions are outlined in Chapter 7.1. Chapter 7 presents EPAS for Maximum Coverage with fairness constraints on both bounded-frequency and $K_{d,d}$-free instances, while Chapter 8 further incorporates matroid constraints on the two instances. The final part of Part II shifts attention to a different parameter, namely the number of satisfied constraints. The full details provided in Chapter 9. This chapter contains an FPT algorithm for CC-MaxSat on general instances, as well as polynomial kernels for both CC-MaxSat and Maximum Coverage on $K_{d,d}$-free instances.

Part III focuses on the second objective. An overview of the results on polynomial-time approximation schemes (PAS) for Maximum Coverage under this objective is presented in Chapter 10.1. Chapter 10.2 develops a PAS for Maximum Coverage on $K_{d,d}$-free instances. The extension of these results to fairness constraints is discussed in Chapter 10.3, followed by hardness of approximation results for this setting in Chapter 10.4.

Part IV introduces and develops a framework for multi-criteria lossy kernelization. The general framework is outlined in Chapter 11. An overview of the lossy kernel for CC-MaxSat with fairness and matroid constraints, parameterized by the solution size, is given in Chapter 12.1. Chapter 12.2 presents a multi-criteria lossy kernel for CC-MaxSat under fairness constraints, and Chapter 12.3 extends this result to include matroid constraints.

Chapter 12 concludes the thesis by outlining several open problems and laying the foundation for future research directions.

Chapter 4

Preliminaries

4.1 Parameterized Complexity

Parameterized Complexity was introduced by Downey and Fellows in the 90s as a framework to cope with computational problems that are NP-Hard. Here, a parameter, which is small compared to the input, is considered. Parameterized Complexity deals with measuring the running time of algorithms in terms of both the input size and the parameter. The following definitions are standard and can be found in [11, 62].

We begin by formally defining what is meant by a parameterized problem.

Definition 1 ([11]). *A parameterized problem is a language $L \in \Sigma^\star \times \mathbb{N}$, where Σ is a fixed, finite alphabet. For an instance (x, k), k is called the parameter.*

The notion of tractability in the parameterized setting is captured by fixed-parameter tractability, which allows exponential dependence on the parameter while maintaining polynomial dependence on the input size.

Definition 2 ([11]). *A parameterized problem L is called fixed-parameter tractable (FPT) if there exists an algorithm $\mathcal{A}$ (called a fixed-parameter algorithm), a computational function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^\star \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k)|(x, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called FPT.*

An important technique closely related to fixed-parameter tractability is *kernelization*. Informally, kernelization is a polynomial-time preprocessing procedure that

reduces a given instance (I, k) of a problem to an equivalent instance (I', k') of the same problem, called a *kernel*, whose size is bounded solely as a function of the parameter. Crucially, the reduced instance preserves the answer to the original problem.

Kernelization provides a structural understanding of parameterized problems by isolating the part of the input that is inherently difficult. Every fixed-parameter tractable problem admits a kernel, although the size of the kernel may be large. A central goal in parameterized complexity is therefore to design *polynomial kernels*, where the size of the reduced instance is bounded by a polynomial function of the parameter.

Definition 3 ([11]). *A kernelization, or simply a kernel, for a fixed-parameter tractable problem Q is an algorithm $\mathcal{A}$ that given an instance (I, k) of Q, works in polynomial time and returns an equivalent instance (I', k') of Q. Moreover, we require that $size_{\mathcal{A}}(k) \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$. Here, $size_{\mathcal{A}}(k)$, also called the output size of the algorithm $\mathcal{A}$ is defined as follows:*

$$size_{\mathcal{A}}(k) = sup\{|I'| + k' : (I', k') = \mathcal{A}(I, k), I \in \Sigma^{\star}\}$$

We define two standard combinatorial tools that will be used in our algorithms, namely universal sets and perfect hash families.

Definition 4 ((n, ℓ)**-universal set**, [11]). *An (n, ℓ)-universal set is a family $\mathcal{U}$ of subsets of $[n]$ such that for every subset $S \subseteq [n]$ of size at most ℓ, the family $\{U \cap S : U \in \mathcal{U}\}$ contains all $2^{|S|}$ subsets of S.*

The following result guarantees that such families of manageable size can be constructed efficiently.

Proposition 2 (Naor et al. [65]). *There is an algorithm that given integers $n, \ell \in \mathbb{N}$, runs in time $2^{\ell}\ell^{\mathcal{O}(\log \ell)} n \log n$, and outputs an (n, ℓ)-universal set of cardinality at most $2^{\ell}\ell^{\mathcal{O}(\log \ell)} \log n$.*

Closely related to universal sets are perfect hash families, which are frequently used in derandomization and color-coding arguments.

Definition 5 ((p, q)-perfect hash family). ([66]) *For non-negative integers p and q, a family of functions $f_1, \ldots, f_t$ from a universe $\mathcal{U}$ of size p to a universe of size q is called a (p, q)-perfect hash family, if for any subset $S \subseteq \mathcal{U}$ of size at most q, there exists $i \in [t]$ such that f_i is injective on S.*

We can construct (p, q)-perfect hash family using the following result.

Proposition 3 ([65, 11]). *There is an algorithm that given $p, q \geq 1$ constructs a (p, q)-perfect hash family of size $e^q q^{\mathcal{O}(\log q)} \log p$ in time $e^q q^{\mathcal{O}(\log q)} p \log p$.*

4.2 Parameterized Approximation

In many optimization problems, exact fixed-parameter algorithms may still be computationally expensive, motivating the study of approximation within the parameterized framework.

Definition 6 (Parameterized Approximation Scheme (PAS)). *A* Parameterized Approximation Scheme (PAS) *is an algorithmic framework that, given a parameter k and a desired approximation accuracy $\varepsilon > 0$, computes a $(1 - \varepsilon)$-approximate solution within a running time of $f(k, \varepsilon) \cdot n^{g(\varepsilon)}$, where reduces$f$ and g are computable functions.*

An important refinement of this notion restricts the dependence on the input size to be polynomial for every fixed parameter and approximation ratio.

Definition 7 (Efficient Parameterized Approximation Scheme (EPAS)). *An* Efficient Parameterized Approximation Scheme (EPAS) *is an algorithmic framework that, given a parameter k and a desired approximation accuracy $\varepsilon > 0$, computes a $(1 - \varepsilon)$-approximate solution within a runtime of $f(k, \varepsilon) \cdot n^{\mathcal{O}(1)}$, , where f is a computable function.*

EPAS ensures polynomial-time dependence on n for every fixed k and ε. For further details and a comprehensive discussion of these parameterized approximation classes, we refer the reader to the survey by Feldmann et al. [67] and the paper by Lokshtanov et al. [62].

4.3 Notations and Conventions

4.3.1 CNF Satisfiability

Let $\Phi = (\mathcal{V}, \mathcal{C})$ be a CNF formula on n variables and m clauses. By $\mathcal{V}_\Phi$ and $\mathcal{C}_\Phi$, we denote the set of variables and clauses in the formula Φ, respectively. A variable v

can appear as a positive literal (v) or as a negative literal ($\overline{v}$) in Φ. Any formula Φ is a conjunction of clauses in $\mathcal{C}$, where each clause is a disjunction of positive and (or) negative literals.

We use G_Φ to denote the clause-variable incidence bipartite graph of Φ. That is, the vertex set of G_Φ is $\mathcal{V} \uplus \mathcal{C}$. For each $v \in \mathcal{V}$ and $C \in \mathcal{C}$, there is an edge between v and C in G_Φ if and only if v or $\overline{v}$ belongs to the clause C.

The positive degree of a variable v, denoted by $\mathsf{deg}^+_\Phi(v)$, is the number of clauses in Φ which contains the literal v. For a subset Y of variables, $N^+_\Phi(Y)$ is the set of clauses in Φ that contains at least one variable from Y positively. We use $\mathsf{deg}^+_\Phi(Y)$ to denote the number $|N^+_\Phi(Y)|$. Similarly, we denote $N^-_\Phi(Y)$ as the set of clauses in Φ that contains at least one variable from Y negatively. For a subset Y of variables $N_\Phi(Y)$ is the set of clauses in Φ that contains at least one variable from Y either positively or negatively, that is the set $N^+_\Phi(Y) \cup N^-_\Phi(Y)$. For a clause $c \in \mathcal{C}_\Phi$, by $\mathsf{var}(c)$, we denote the set of variables contained in c. For a set of clauses $C \subseteq \mathcal{C}_\Phi$, $\mathsf{var}(C) = \bigcup_{c \in C} \mathsf{var}(c)$. For a clause c and variable $u \in \mathcal{V}_\Phi$, by $c \setminus \{u\}$, we denote the clause c after removing the literal corresponding to u from c. For example let $c = u \vee v \vee w$, then $c \setminus \{u\} = v \vee w$. If $u \notin \mathsf{var}(c)$, then $c \setminus \{u\}$ is simply the clause c. We remove the subscript Φ, if it is clear from the context.

An *assignment* to a CNF formula Φ is a function $\beta : \mathcal{V} \to \{0, 1\}$. If a variable is assigned 1(or 0), then we also say that the variable has been assigned true(or false). The *weight* of an assignment β is the number of variables that have been assigned 1. By $\beta^{-1}(1)$(or $\beta^{-1}(0)$), we denote the set of variables that have been assigned 1 (0). Alternatively, we use $T(\Psi)$ and $F(\Psi)$, to denote the set of variables assigned 1 and 0 by an assignment Ψ, respectively. For a variable subset Z, the assignment β_Z is defined as follows. For each variable x, $\beta_Z(x) = 1$ if and only if $x \in Z$. An *all zero* assignment $\emptyset$ or $\beta_\emptyset$ assigns false value to every variable of the input formula.

For a formula Φ, we use $\mathsf{OPT}_k(\Phi)$ to denote the maximum number of clauses in Φ that can be satisfied by an assignment of weight at most k. We alternatively use $\mathsf{OPT}^{\mathrm{SAT}}_{\mathsf{k}}(\Phi)$ to denote the maximum number of clauses in Φ that can be satisfied by an assignment of weight at most k, or just $\mathsf{OPT}^{\mathrm{SAT}}_{\mathsf{k}}$ when the formula is clear from context. For an assignment β, $\mathsf{pos}_\Phi(\beta)$ is the set of clauses in Φ that contains at least one variable from $\beta^{-1}(1)$ as a positive literal. That is, $\mathsf{pos}_\Phi(\beta)$ is the set of clauses satisfied by the variables set to true by β. We say that a variable subset Y is an *optimum solution* to (Φ, k) if $|Y| \leq k$ and the number of clauses satisfied by β_Y is equal to $\mathsf{OPT}_k(\Phi)$.

4.3.2 Graph

For a graph G, we denote its vertex set by $V(G)$ and its edge set by $E(G)$. We define a bipartite graph $G = (A \uplus B, E)$ as a graph where $V(G)$ can be partitioned into two parts A and B, where each part is an independent set. For two sets A and B, $A \backslash B$ denotes the set difference of A and B, i.e., the set of elements in A, but not in B. For a bipartite graph $G = (A \uplus B, E)$, and a subset $X \subseteq V(G)$, we denote by $\Delta(X)$ the maximum degree of a vertex in X. The set of neighbors of a vertex $v \in V(G)$ in the graph G is denoted by $N_G(v)$, or just $N(v)$ when the graph is clear from context, and is also called the open neighborhood of v. The closed neighborhood of v is denoted by $N_G[v] = N_G(v) \cup \{v\}$. When the graph is clear from context, we just call it $N[v]$. For a subset of vertices $Y \subseteq V(G)$, the closed neighborhood of Y is $N_G[Y] = \cup_{v \in Y} N_G[v]$ and the open neighborhood of Y is $N_G(Y) = N_G[Y] \backslash Y$. For a vertex $v \in V(G)$, we denote by $\mathsf{deg}_G(v)$, or just $\mathsf{deg}(v)$, the number of edges incident on v. The graph $K_{d,d}$ denotes the complete bipartite graph with each bipartition containing d vertices. A bipartite graph $G = (A, B, E)$ is $K_{d,d}$-free when G excludes $K_{d,d}$ as an induced subgraph (no set of d vertices in A together have d common neighbors in B). For a set $X \subseteq V(G)$, E_X denotes the set of edges with at least one endpoint in X. When $X = \{u\}$, then for ease of notation we write E_u. Let H be an induced subgraph of G. We define some terminology w.r.t. the graph $H = G[A' \uplus B']$, where $A' \subseteq A$, and $B' \subseteq B$. For a vertex $v \in A'$, and a color j, let $N_j^H(v) \subseteq B' \cap B_j$ denote the set of neighbors of v of color j. More formally, let $N_j^H(u) \coloneqq \{\mathfrak{s} \in B' : f(\mathfrak{s}) = j \text{ and } (u, \mathfrak{s}) \in E(H)\}$. Furthermore, for a subset $S \subseteq A$, let $N_j(S) \coloneqq \bigcup_{v \in S} N_j(v)$. We also define $d_j^H(v) \coloneqq |N_j^H(v)|$. We call $d_j^H(v)$ as j-degree of v in the graph H. We may omit the superscript from these notations when $H = G$. Finally, for a vertex $v \in A'$, we define $H \setminus\setminus v$ as the graph obtained by deleting $N^H[v]$, i.e., v and all of its neighbors in B'. Consider an instance $\mathcal{I} = (G, [r], f, t, k)$ of PCCDS. For a vertex $\mathfrak{s} \in B$, $f_{\setminus \mathfrak{s}}$ is a restriction of f to the set $B \setminus \{\mathfrak{s}\}$.

4.3.3 Sets

We use $\mathbb{N}$ and $\mathbb{Q}$ to denote the sets of natural numbers and rational numbers, respectively. Additionally, we define $\mathbb{Q}^+$ to represent the set of positive rational numbers, that is $\mathbb{Q}^+ = \{q \mid q \in \mathbb{Q}, q > 0\}$. For a number $\alpha \in \mathbb{N}$, we use $[\alpha]$, to denote the set $\{1, \ldots, \alpha\}$. Similarly, for $a, b \in \mathbb{N}, b \geq a$, we use $[a, b]$ to denote the set $\{a, a+1, \ldots, b\}$. For a set B, the power set is the set of all possible subsets of

B, including $\emptyset$ and B itself. We denote the power set of B as 2^B. A *set system* $(\mathcal{U}, \mathcal{F})$ consists of a universe $\mathcal{U} \coloneqq \{u_1, \ldots, u_n\}$ of n elements and a family $\mathcal{F} \subseteq 2^{\mathcal{U}}$ containing m subsets of $\mathcal{U}$. For a coverage requirement function t (resp. variations such as $t', \widetilde{t}$), and a color j, we shorten $t(j)$ to t_j (resp. $t'_j, \widetilde{t}_j$).

4.3.4 Matroids and Representative Sets

Matroids and Representative sets: A matroid $\mathcal{M} = (U, I)$, consists of a finite universe U and a family I of sets over U that satisfies the following three properties:

1. $\emptyset \in I$,
2. if $A \in I$ and $B \subseteq A$, then $B \in I$,
3. if $A \in I$ and $B \in I$ and $|A| < |B|$ then there is an element $b \in B \setminus A$ such that $A \cup \{b\} \in I$.

Each set $S \in I$ is called an independent set and an inclusion-wise maximal independent set is known as a basis of the matroid. The rank of a set $P \subseteq U$ is defined as the size of the largest subset of P that is independent, and is denoted by $\mathsf{rank}(P)$. Note that $\mathsf{rank}(P) \leq \mathsf{rank}(Q)$ if $P \subseteq Q$. From the third property of matroids, it is easy to observe that every inclusion-wise maximal independent set has the same size; this size is referred to as the *rank* of the matroid. For any $P \subseteq U$, define the closure of P, $\mathsf{cl}(P) \coloneqq \{x \in U : \mathsf{rank}(P \cup \{x\}) = \mathsf{rank}(P)\}$. Note that $\mathsf{cl}(P) \subseteq \mathsf{cl}(Q)$ if $P \subseteq Q$.

A matroid is linear (representable) if it can be defined using linear independence, that is for any such matroid $\mathcal{M} = (U, I)$, one can assign to every $e \in U$ a vector v_e over some field (where the different elements of the universe should all be over the same field $\mathbf{F}$ and have the same dimension) such that a set $S \subseteq U$ is in I if and only if the set $\{v_e : e \in S\}$ forms a linearly independent set of vectors.

We next recall the notion of *contraction* in matroids. Let $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ be a matroid, and let $Z \subseteq \mathcal{E}$ be a subset with X denoting a basis of Z. We define the family $\mathcal{I}_Z$ as follows:

$$\mathcal{I}_Z \coloneqq \{I \subseteq \mathcal{E} \setminus Z \mid I \cup X \in \mathcal{I}\}. \tag{4.1}$$

Let $\mathcal{M}/Z = (\mathcal{E} \setminus Z, \mathcal{I}_Z)$ denote the set system thus obtained. It is well known that $\mathcal{M}/Z$ is itself a matroid.

Proposition 4 ([68]). *$\mathcal{M}/Z = (\mathcal{E} \setminus Z, \mathcal{I}_Z)$ is a matroid.*

Furthermore, when Z is a singleton set, i.e., $Z = \{v\}$, we write $\mathcal{M}/v = (\mathcal{E} \setminus \{v\}, \mathcal{I}_v)$ instead of $\mathcal{M}/\{v\} = (\mathcal{E} \setminus \{v\}, \mathcal{I}_{\{v\}})$.

Let $\mathcal{M} = (U, I)$ be a matroid of rank k. Then, for any $0 \leq k' \leq k$, one can define a truncated version of the matroid as follows: $\mathcal{M}_{k'} = (U, I_{k'})$, where $I_{k'} = \{S \in I : |S| \leq k'\}$. It is easy to verify that the contraction as well as truncation operation results in a matroid. Furthermore, given a linear representation of a matroid $\mathcal{M}$, a linear representation of the matroid resulting from contraction/truncation can be computed in (randomized) polynomial time [36, 69]. Alternatively, given an oracle access to the original matroid $\mathcal{M}$, one can simulate the oracle access to the contracted/truncated matroid with at most a linear overhead.

Notice that given the independence oracle for the original matroid $\mathcal{M}$, one can trivially simulate the independence oracle for the truncated matroid by additionally checking the size bound.

We call a family of sets of size p as p-family. Next, we state the following crucial definition of representative families.

Definition 8 ([37, 11]). *Let $\mathcal{M} = (U, I)$ be a matroid and $\mathcal{A}$ be a p-family of independent subsets of U. A subfamily $\mathcal{A}' \subseteq \mathcal{A}$ is said to q-represent $\mathcal{A}$ w.r.t. $\mathcal{M}$ if for every set B of size q such that there is an $A \in \mathcal{A}$ such that $A \cup B$ is an independent set, there is an $A' \in \mathcal{A}'$ such that $A' \cup B$ is an independent set. If $\mathcal{A}'$ q-represents $\mathcal{A}$ w.r.t. $\mathcal{M}$, we write $\mathcal{A}' \subseteq^q_{rep,\mathcal{M}} \mathcal{A}$. If the matroid $\mathcal{M}$ is obvious from the context, then we simply write $\mathcal{A}' \subseteq^q_{rep} \mathcal{A}$.*

Proposition 5 ([37, 11]). *There is an algorithm that, given a matrix M over a field $GF(s)$, representing a matroid $\mathcal{M} = (U, F)$ of rank k, a p-family $\mathcal{A}$ of independent sets in $\mathcal{M}$, and an integer q such that $p + q = k$, computes a q-representative family $\mathcal{A}' \subseteq^q_{rep} \mathcal{A}$ of size at most $\binom{p+q}{p}$ using at most $\mathcal{O}(|\mathcal{A}|(\binom{p+q}{p})p^\omega + (\binom{p+q}{p})^{\omega-1})$ operations over $GF(s)$.*

In the following lemma, we show that $\mathcal{A}' \subseteq^{k-1}_{rep} \mathcal{A}$ can be computed in polynomial time using oracle access to $\mathcal{M}$.

Lemma 1. *Let $\mathcal{M} = (U, I)$ be a matroid of rank k given via oracle access, and let $\mathcal{A}$ be a 1-family of subsets of U. Then, $\mathcal{A}' \subseteq^{k-1}_{rep} \mathcal{A}$ can be computed in polynomial time, where $|\mathcal{A}'| \leq k$.*

Proof. Let $A = \{x : \{x\} \in \mathcal{A}\}$ be the set of underlying elements corresponding to the sets of $\mathcal{A}$. We compute an inclusion-wise maximal subset $R \subseteq A$ that is independent in $\mathcal{M}$. Note that this subset can be computed using $O(|U|k)$ queries, where $k = \mathsf{rank}(\mathcal{M})$. We claim that $\mathcal{A}' = \{\{y\} : y \in R\}$ is a $(k-1)$-representative set of $\mathcal{A}$.

Consider any $u \in A$ and $X \subseteq U$ of size k such that $u \in X$ and $X \in I$. We will show that there exists some $u' \in R$ such that $X \setminus \{u\} \cup \{u'\} \in \mathcal{I}$. Since R is an inclusion-wise maximal independent subset of A, it follows that $u \in \mathsf{cl}(R) \subseteq \mathsf{cl}(R \cup (X \setminus \{u\}))$. This means that $\mathsf{rank}(R \cup (X \setminus \{u\})) = \mathsf{rank}(R \cup (X \setminus \{u\}) \cup \{u\}) = \mathsf{rank}(R \cup X)$. However, $\mathsf{rank}(R \cup X) \geq \mathsf{rank}(X)$. Therefore, we obtain that $\mathsf{rank}(R \cup (X \setminus \{u\})) \geq \mathsf{rank}(X)$. This implies that there exists some $u' \in R$ such that $X \setminus \{u\} \cup \{u'\} \in I$. □

We also state the following lemma.

Lemma 2. *Let $\mathcal{M} = (U, I)$ be a matroid of rank k, let $\mathcal{A}$ be a 1-family of subsets of U, and let $\mathcal{A}' \subseteq^{k-1}_{rep,\mathcal{M}} \mathcal{A}$. Then, for any $1 \leq k' \leq k$, $\mathcal{A}' \subseteq^{k'-1}_{rep,\mathcal{M}_{k'}} \mathcal{A}$, where recall $\mathcal{M}_{k'} = (U, I_{k'})$ is the k'-truncation of $\mathcal{M}$.*

Proof. Consider 1-families $\mathcal{A}, \mathcal{A}'$ as in the statement of the lemma. Fix $1 \leq k' \leq k-1$. Consider any $S \subseteq U$ of size $k'-1$, such that (i) $S \in I_{k'-1} \subseteq I$, and (ii) there exists some $\{u\} \in \mathcal{A}$ such that, $u \notin S$, and (iii) $S \uplus \{u\} \in I_{k'} \subseteq I$. We want to show that there exists some $\{u'\} \in \mathcal{A}'$ such that $u' \notin S$, and $S \uplus \{u'\} \in I_{k'} \subseteq I$.

To this end, note that due to matroid properties, $S_1 \coloneqq S \uplus \{u\}$ can always be extended to a basis of $\mathcal{M}$. Consider an arbitrary such basis, say S^*, such that $S_1 \subseteq S^*$. Note that $|S^*| = k > k-1 \geq k'-1$. Therefore, S^* contains a subset $\tilde{S}$ of size $k-1$, such that $\tilde{S} \in I_{k-1}$, and $u \notin \tilde{S}$. Further, since $\tilde{S} \uplus \{u\} \in I$, and $\mathcal{A}' \subseteq^{k-1}_{rep,\mathcal{M}} \mathcal{A}$, it follows that there exists some $\{u'\} \in \mathcal{A}'$, such that $u' \notin \tilde{S}$, and $\tilde{S} \uplus \{u'\} \in I$. Now, note that $S \uplus \{u'\} \subseteq \tilde{S} \uplus \{u\}$, and due to matroid properties $S \uplus \{u'\} \in I_{k'}$. This completes the proof that $\mathcal{A}' \subseteq^{k'-1}_{rep,\mathcal{M}_{k'}} \mathcal{A}$. □

Part II: Optimizing Constraints

Chapter 5

EPAS for Satisfiability and Partial Coverage on $K_{d,d}$-free instances

This chapter addresses our first objective. Building on the motivations developed earlier for studying EPAS parameterized by the solution size on structured instances, we design efficient parameterized approximation schemes (EPAS) for CC-MaxSat and Maximum Coverage on $K_{d,d}$-free instances.

We show how combinatorial properties of $K_{d,d}$-free incidence graphs of the input can be leveraged to either identify a variable (or set) that must participate in an optimal solution or extract a near-optimal solution, thereby yielding the desired EPAS.

5.1 EPAS for Max k-weight SAT on $K_{d,d}$-free formulas

We begin with the EPAS for CC-MaxSat on $K_{d,d}$-free formulas parameterized by the solution size k.

Let (Φ, k) be the input instance of CC-MaxSat such that G_Φ is $K_{d,d}$-free, and $\varepsilon > 0$ be the error parameter. For an assignment β, let $\mathsf{pos}_\Phi(\beta)$ be the set of clauses in Φ that contains at least one variable from $\beta^{-1}(1)$ positively. That is, $\mathsf{pos}_\Phi(\beta)$ is the set of clauses satisfied by the variables assigned 1 by β. A variable subset $Y \subseteq \mathcal{V}_\Phi$ is called a solution to (Φ, k) if $|Y| \leq k$. The number of clauses satisfied by the solution Y is the number of clauses satisfied by the assignment $\beta_Y \colon \mathcal{V}_\Phi \to \{0, 1\}$ which is defined as follows. For any $x \in \mathcal{V}_\Phi$, $\beta_Y(x) = 1$ if and only if $x \in Y$. For

a formula Φ, we use $\mathsf{OPT}_k(\Phi)$ to denote the maximum number of clauses in Φ that can be satisfied by an assignment of weight at most k. We say that S is an *optimum solution* to (Φ, k) if $|S| \leq k$ and the number of clauses satisfied by β_S is equal to $\mathsf{OPT}_k(\Phi)$.

The key idea to design the algorithm is the following. Let S be an optimum solution and let $\mathsf{pos}_\Phi(\beta_S) = t$. We prove that there exist functions f and g such that if $t \geq f(k, d, \varepsilon)$, then the set X of $g(k, d, \varepsilon)$ variables with highest positive degrees (i.e., the number of clauses it appears in positively) has the following property. Either $X \cap S \neq \emptyset$ or there is a variable subset $Q \subseteq X$ of size at most k such that $\mathsf{pos}_\Phi(\beta_Q) \geq (1-\varepsilon)t$. Using this result, we prove that the set $X^\star$ of $(1+\frac{1}{\varepsilon})g(k, d, \varepsilon)$ variables with highest positive degrees has the following property. Either $X^\star \cap S \neq \emptyset$ or $X^\star$ contains a $(1-\varepsilon)$-approximate solution to (Φ, k). If $X^\star \cap S \neq \emptyset$, then we guess a variable from $X^\star \cap S$, reduce the formula Φ and recursively solve with the parameter $k-1$. Otherwise, by considering all the subsets of $X^\star$ of size at most k, we get a $(1-\varepsilon)$-approximate solution to (Φ, k).

If $t < f(k, d, \varepsilon)$, then we solve the problem exactly using random separation. Suppose $\varepsilon\mathsf{OPT}_k(\Phi) \geq f(k, d, \varepsilon)$, then $\emptyset$ (i.e., the assignment that assigns all variables 0) is a $(1-\varepsilon)$-approximate solution to (Φ, k). Otherwise, $\mathsf{OPT}_k(\Phi) \leq \frac{1}{\varepsilon} f(k, d, \varepsilon)$ and in this case we design an FPT algorithm parameterized by k, d, ε for the problem using random separation.

For clarity of presentation, we assume that $\frac{1}{\varepsilon}$ is an integer. Otherwise, we may replace ε with the largest number ε' less than ε with the property that $\frac{1}{\varepsilon'}$ is an integer. For simplicity, we omit all floor and ceiling signs whenever these are not crucial.

Next, we define a notion of *positively* (ε, t)*-good set* which we crucially use in our algorithms.

Definition 9. *Let Φ be a CNF formula, and let $t \in \mathbb{N}$ and $\varepsilon > 0$. A subset of variables S of size at most k is said to be* positively (ε, t)-good *in Φ if at least $(1-\varepsilon)t$ clauses in Φ contain some variable from S appearing positively.*

Our next lemma shows that if we select top $f(k, d, \varepsilon)$ variables with the highest positive degrees for some function f, then it contains a positively (ε, t)-good set.

Lemma 3. *Let (Φ, k) be an instance of* CC-MaxSat *such that G_Φ is $K_{d,d}$-free and $\varepsilon > 0$. Further, assume that there exists an assignment β of weight at most k such that $|\mathsf{pos}_\Phi(\beta)| = t$. Let $q = k + \frac{4dk}{\varepsilon}$ and $X = \{v_1, \ldots, v_q\}$ be the first q variables with*

the highest positive degrees. If $\frac{\varepsilon t}{2k} \geq q^d d$*, then* X *contains a positively* (ε, t)*-good set in* Φ*.*

Proof. Let β be an assignment of weight at most k such that $|\mathsf{pos}_\Phi(\beta)| = t$. Further, we assume that the variables in the set X are ordered as follows: $\mathsf{deg}^+_\Phi(v_1) \geq \mathsf{deg}^+_\Phi(v_2) \geq \cdots \geq \mathsf{deg}^+_\Phi(v_q)$. Observe that $\beta^{-1}(1)$ may not be fully contained in X. We describe a process to construct a positively (ε, t)-good set in Φ contained in X starting with $(\beta^{-1}(1) \cap X)$. Indeed, let $W = \beta^{-1}(1)$ and $\{z_1, \ldots, z_\ell\} = W \setminus X$. Next, we find vertices $w_1, \ldots, w_\ell \in X \setminus W$ which we add to $W \cap X$ in order to construct a positively (ε, t)-good set in Φ contained in X. This is done as follows. Let $W_0 = W$. For each $i \in [\ell]$, let h_i be the *least integer* such that there is a vertex w_i in $X \setminus W_{i-1}$ with the following property.

$$\mathsf{deg}^+(W_{i-1}) = \mathsf{deg}^+((W_{i-1} \cup \{w_i\}) \setminus \{z_i\}) + h_i \tag{5.1}$$

That is, w_i is the best available choice to exchange with z_i. This implies that for any $u \in X \setminus W_{i-1}$,

$$\mathsf{deg}^+(W_{i-1}) \geq \mathsf{deg}^+((W_{i-1} \cup \{u\}) \setminus \{z_i\}) + h_i. \tag{5.2}$$

Then, we set $W_i = (W_{i-1} \cup \{w_i\}) \setminus \{z_i\}$. That is,

$$\mathsf{deg}^+(W_{i-1}) = \mathsf{deg}^+(W_i) + h_i \tag{5.3}$$

Observe that if $h_i \leq 0$, then W_i is at least as good as W_{i-1}.

Claim 1. W_ℓ *is positively* (ε, t)*-good in* Φ*.*

Proof. Since $W = \beta^{-1}(1)$ and $|\mathsf{pos}_\Phi(\beta)| = t$, we have that $\mathsf{deg}^+(W) = t$. Summing (Equation 5.3) for all $i \in [\ell]$, we get $\sum_{i=1}^{\ell} \mathsf{deg}^+(W_{i-1}) = \sum_{i=1}^{\ell} (\mathsf{deg}^+(W_i) + h_i)$. This implies that $\mathsf{deg}^+(W) - \mathsf{deg}^+(W_\ell) = \sum_{i=1}^{\ell} h_i$. Let $I = \{i \in [\ell] \,:\, h_i > 0\}$. To prove the claim, it is enough to show that $\sum_{i \in I} h_i \leq \varepsilon t$.

For each $i \in I$, we construct a bipartite graph $H_i = (L_i \uplus R_i, E_i)$, where $L_i = X \setminus W_i$ and $R_i = N^+(W_{i-1} \setminus \{z_i\})$. There is an edge between $v \in L_i$ and $C \in R_i$ if and only if the literal v belongs to the clause C. To better understand why we construct this bipartite graph, we set L_i and R_i as we do because we want a lower bound on $\mathsf{deg}_{H_i}(v)$ for each $v \in L_i$. That is, we want to calculate how many clauses a positive literal $v \in X \setminus W_i$ occurs in among the clauses that variables in $W_{i-1} \setminus \{z_i\}$ already

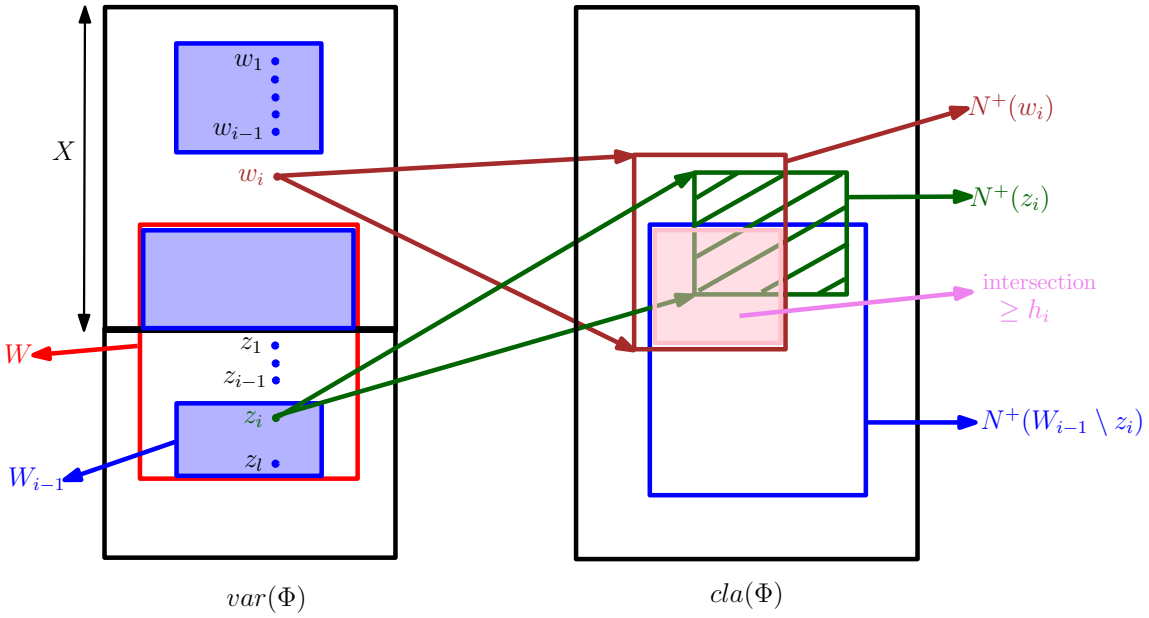

Figure 5.1: Figure depicting at the i^{th} iteration, $deg^+(v)$ in $N^+(W_{i-1} \setminus \{z_i\})$ is at least h_i for any $v \in X \setminus W_i$. Any variable in $X \setminus W_i$ appears positively in at least h_i clauses that are already satisfied positively by $W_{i-1} \setminus \{z_i\}$. In particular, $deg^+(w_i)$ in $N^+(W_{i-1} \setminus \{z_i\})$ is exactly h_i.

occur in positively. We show that $\mathsf{deg}_{H_i}(v) \geq h_i$, the loss encountered at the i^{th} step. This gives us an upper bound on each h_i for $i \in [I]$, and hence, an upper bound on $\sum_{i \in I} h_i$, as we see next. Figure 5.1 depicts how the set of clauses satisfied positively by any $v \in (X \setminus W_i) \cup \{w_i\}$ interacts with the set of clauses satisfied positively by $W_{i-1} \setminus \{z_i\}$ pictorially. We show that the interaction is at least h_i.

Next, we prove that if $|R_i| > 2t$, then W_ℓ is positively (ε, t)-good. Since $\mathsf{deg}^+(W) = t$, $\mathsf{deg}^+(W_{i-1} \setminus (\{z_i\} \cup X)) \leq t$. Hence, if $|R_i| > 2t$, then $\mathsf{deg}^+(X \cap W_{i-1}) > t$. This implies that $(X \cap W_{i-1}) \subseteq W_\ell$ is positively (ε, t)-good for Φ. Now on, we assume that $|R_i| \leq 2t$. Equation 5.2 implies that for every $v \in L_i$, $\mathsf{deg}_{H_i}(v) \geq h_i$. Suppose not, then there exists a $\hat{v} \in L_i$ such that $\mathsf{deg}_{H_i}(\hat{v}) \leq h_i - 1$. Then,

$$
\begin{aligned}
|N^+(W_i)| &\geq |N^+(W_{i-1} \setminus \{z_i\})| + |N^+(\hat{v})| - |N^+(\hat{v}) \cap N^+(W_{i-1} \setminus \{z_i\})| \\
&\geq |N^+(W_{i-1})| - |N^+(z_i)| + |N^+(\hat{v})| - |N^+(\hat{v}) \cap N^+(W_{i-1} \setminus \{z_i\})| \\
&\geq |N^+(W_{i-1})| - |N^+(\hat{v}) \cap N^+(W_{i-1} \setminus \{z_i\})| \qquad \text{(Because } \hat{v} \in X \setminus W_i \text{ and } z_i \notin X\text{)} \\
&\geq |N^+(W_{i-1})| - h_i + 1 \quad \text{(Because } |N^+(\hat{v}) \cap N^+(W_{i-1} \setminus \{z_i\})| = \mathsf{deg}_{H_i}(\hat{v}) \leq h_i - 1\text{)}
\end{aligned}
$$

which contradicts Equation 5.2.

Thus, the total number of edges in H_i is at least $h_i|L_i|$. Now, we give an upper bound on the total number of edges. Since H_i is a subgraph of G_Φ, H_i is $K_{d,d}$-free.

For any fixed subset L_i' of L_i of size d, there are at most $d-1$ elements whose neighborhood contains L_i'. This implies that there are at most $|L|^d d$ vertices in R_i with degree at least d. Thus, the total number of edges in H_i is upper bounded by

$$|L_i|^d \cdot d \cdot |L_i| + |R_i|(d-1).$$

From the lower bound and upper bound on $|E(H_i)|$, we get

$$\begin{aligned}
h_i|L_i| &\leq |L_i|^d \cdot d \cdot |L_i| + |R_i|(d-1) \\
&\leq |L_i|^d \cdot d \cdot |L_i| + 2td \\
h_i &\leq |L_i|^d \cdot d + \frac{2td}{|L_i|} \\
&\leq |L_i|^d \cdot d + \frac{\varepsilon t}{2k} && \text{(Because } |L_i| \geq q - k = \frac{4kd}{\varepsilon}\text{)} \\
&\leq q^d \cdot d + \frac{\varepsilon t}{2k} && \text{(Because } |L_i| \leq q\text{)} \\
&\leq \frac{\varepsilon t}{2k} + \frac{\varepsilon t}{2k} && \text{(Because } q^d d \leq \frac{\varepsilon t}{2k}\text{)} \\
&\leq \frac{\varepsilon t}{k}
\end{aligned}$$

This implies that $\sum_{i \in I} h_i \leq \varepsilon t$. □

This completes the proof of the lemma. □

Let S be an optimum solution and let $|\mathsf{pos}_\Phi(\beta_S)| = t$. The next lemma shows that if we select top $g(k, d, \varepsilon)$ variables with the highest positive degrees for some g, then it either contains a variable from S (that is, it will be assigned positively) or it already contains an approximate solution $Y \subseteq X$ (the number of clauses satisfied by β_Y is at least $(1-\varepsilon)\mathsf{OPT}_k(\Phi)$). In the latter case, we are done. However, in the former case we could branch on the top $g(k, d, \varepsilon)$ variables by assigning it to true. Note that in this branching, the *parameter* k drops, and hence, the depth of the tree is bounded by k, leading to the desired EPAS for the problem.

Remark 1. *Notice that Definition 9 is equivalent to stating that, given a bipartite graph G with $V(G) = A \uplus B$, a subset of A of size at most k is said to be "positively (ε, t)-good" if it has at least $(1-\varepsilon)t$ neighbors in B. Lemma 3 captures the idea that the set of q highest-degree vertices in A contains a positively (ε, t)-good set, provided there exists a set of size at most k in A with t neighbors in B and $\frac{\varepsilon t}{2k} \geq q^d d$.*

Lemma 4. *Let (Φ, k) be an instance of* CC-MaxSat *such that G_Φ is $K_{d,d}$-free and $\varepsilon > 0$. Let $q = k + \frac{4dk}{\varepsilon}$ and $q^\star = \left(1 + \frac{1}{\varepsilon}\right) q$. Let $X = \{v_1, \ldots, v_{q^\star}\}$ be the first*

$q^\star$ variables with the highest positive degrees. Let S be an optimum solution and let $|\mathsf{pos}_\Phi(\beta_S)| = t$. Suppose $\frac{\varepsilon t}{2k} \geq q^d d$. Then, either $S \cap X \neq \emptyset$ or there is a solution $Y \subseteq X$ such that the number of clauses satisfied by β_Y is at least $(1-\varepsilon)\mathsf{OPT}_k(\Phi)$.

Proof. We assume that the variables in the set X are ordered as follows:

$$\mathsf{deg}^+_\Phi(v_1) \geq \mathsf{deg}^+_\Phi(v_2) \geq \cdots \geq \mathsf{deg}^+_\Phi(v_{q^\star}).$$

If $S \cap X \neq \emptyset$, then we are done. From now on, we assume that $S \cap X = \emptyset$. For each $i \in [1 + \frac{1}{\varepsilon}]$, let $X_i = \{v_1, v_2, \ldots, v_{iq}\}$. For each $i \in \{0, \ldots 1/\varepsilon\}$, let Φ_i be the CNF-formula defined as follows. From each clause in Φ delete the variables from X_i, if it appears positively. The resulting formula is denoted by Φ_i.

Let $\Phi_0 = \Phi$ and $X_0 = \emptyset$. For each $i \in [1 + \frac{1}{\varepsilon}]$, let $Z_i = X_i \setminus X_{i-1}$. Notice that for all $i \in \{0, 1, \ldots, \frac{1}{\varepsilon}\}$, Z_{i+1} is the set of q variables with the highest positive degrees in Φ_i.

Since $S \cap X = \emptyset$, S is an optimum solution for Φ_i for all $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$. Moreover, for all $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$, $|\mathsf{pos}_{\Phi_i}(\beta_S)| = t$ and $\mathsf{OPT}_k(\Phi_i) = \mathsf{OPT}_k(\Phi)$. Further, note that an assignment satisfying at least $(1-\varepsilon)\mathsf{OPT}_k(\Phi_i)$ clauses of Φ_i, also satisfies $(1-\varepsilon)\mathsf{OPT}_k(\Phi_i)$ clauses of Φ.

Observe that the set Z_{i+1} has size q and $\frac{\varepsilon t}{2k} \geq q^d d$. Thus, we can apply Lemma 3 on (Φ_i, k), $i \in \{0, 1, \ldots, \frac{1}{\varepsilon}\}$, and obtain a subset $Q_{i+1} \subseteq Z_{i+1}$ of size at most k such that Q_{i+1} is positively (ε, t)-good in Φ_i. Figure 5.2 clearly depicts X_i, Z_i, Q_i for each $i \in [1 + \frac{1}{\varepsilon}]$ and S in $\mathcal{V}_\Phi$.

Claim 2. *There is an integer $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$ such that the number of clauses satisfied by $\beta_{Q_{i+1}}$ is at least $(1-\varepsilon)\mathsf{OPT}_k(\Phi_i)$.*

Proof. Let $\widehat{t} = \mathsf{OPT}_k(\Phi) - t$. Since $\mathsf{OPT}_k(\Phi_i) = \mathsf{OPT}_k(\Phi)$, we have that the *number* $\widehat{t} = \mathsf{OPT}_k(\Phi_i) - t$ for all $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$. Since S is an optimum solution for (Φ_i, k) and $|\mathsf{pos}_{\Phi_i}(\beta_S)| = t$, the number of clauses in Φ_i satisfied *purely* because of the negative literals in the assignment β_S is equal to $\widehat{t}$ for all $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$. Suppose there is an index $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$ such that the number of clauses satisfied only because of the negative literals in the assignment $\beta_{Q_{i+1}}$ is at least $(1-\varepsilon)\widehat{t}$, then combined with the fact that Q_{i+1} is positively (ε, t)-good in Φ_i, the number of clauses satisfied by $\beta_{Q_{i+1}}$ is at least $(1-\varepsilon)\mathsf{OPT}_k(\Phi_i)$.

If the case described in the preceding paragraph does not occur, we will reach a contradiction as explained below. In this case, for every $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$, the number

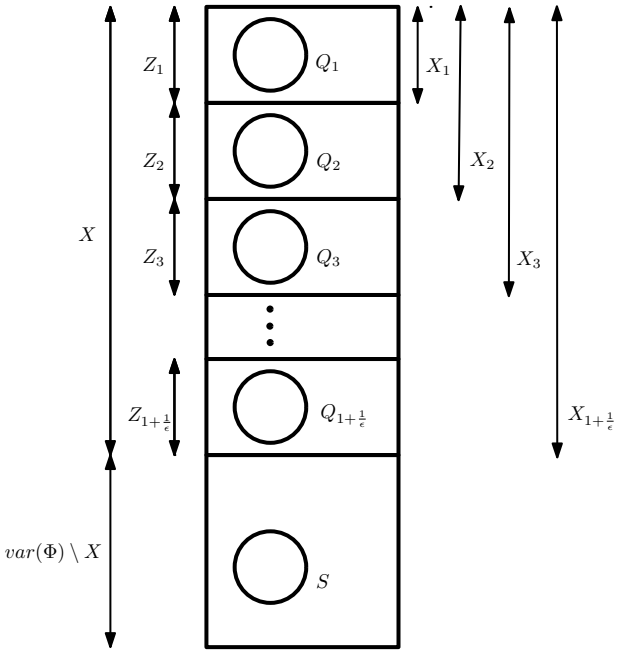

Figure 5.2: Figure depicting the partition of $\mathcal{V}_\Phi$ into X and $\mathcal{V}_\Phi \setminus X$. X_i denotes the set of first iq variables in the set X which is ordered non-increasingly according to their positive degrees, for each $i \in [1 + \frac{1}{\varepsilon}]$. Z_{i+1} denotes the q variables with the highest positive degrees and Q_{i+1} denotes a $(1-\varepsilon)$-approximate solution of size k in the formula Φ_i, for each $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$.

of clauses satisfied only because of the negative literals in the assignment $\beta_{Q_{i+1}}$ is strictly less than $(1-\varepsilon)\widehat{t}$. For a set of variables $Y \in \mathcal{V}_\Phi$, let $neg_\Phi(\beta_Y)$ be the set of clauses in Φ satisfied only because of the negative literals in the assignment β_Y and do not contain any variables from S as a positive literal. Then, $|neg_{\Phi_i}(\beta_{Q_{i+1}})| < (1-\varepsilon)\widehat{t}$. Also, since $S \cap X = \emptyset$, the number of clauses satisfied only because of the negative literals in the assignment β_S is equal to $\widehat{t}$, i.e., $|neg_{\Phi_i}(\beta_S)| = \widehat{t}$. Since, $|neg_{\Phi_i}(\beta_{Q_{i+1}})| < (1-\varepsilon)\widehat{t}$, we have that $|neg_{\Phi_i}(\beta_{Q_{i+1}}) \cap neg_{\Phi_i}(\beta_S)| < (1-\varepsilon)\widehat{t}$. This implies $|neg_{\Phi_i}(\beta_S) \setminus neg_{\Phi_i}(\beta_{Q_{i+1}})| \geq \varepsilon\widehat{t}$. That is, the number of clauses that contain only *negative literals of the variables* from the set Q_{i+1} and *no positive literals of the variables* from S, is at least $\varepsilon\widehat{t}$ for all $i \in \{0, \ldots, \frac{1}{\varepsilon}\}$. Figure 5.3 pictorially depicts what happens in case of a contradiction.

In essence, we have identified a private set of clauses corresponding to each Q_i, which is only satisfied when the variables in Q_i are set to false. Since $\{Q_1, \ldots, Q_{1+\frac{1}{\varepsilon}}\}$ are pairwise disjoint, the number of clauses that contain only negative literals of the variables from the set $\bigcup_{j \in [1+\frac{1}{\varepsilon}]} Q_j$ and no positive literals of the variables from S, is *strictly* more than $\widehat{t}$. This implies that the number of clauses satisfied by β_S in Φ, is strictly more than $t + \widehat{t} = \mathsf{OPT}_k(\Phi)$, which is a contradiction. □

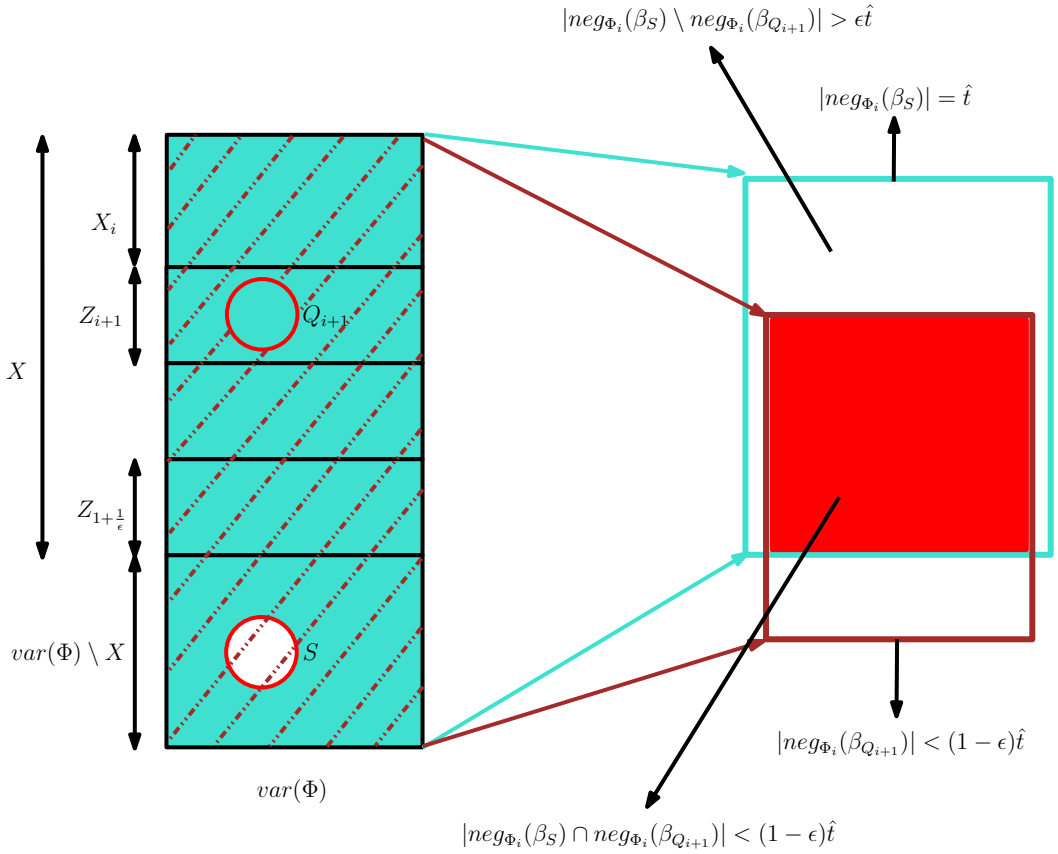

Figure 5.3: Figure depicting the contradiction in Claim 2. The turquoise set depicts the set of variables that can occur as negative literals in β_S and the shaded brown set depicts the set of variables that can occur as negative literals in $\beta_{Q_{i+1}}$. Since, $|neg_{\Phi_i}(\beta_{Q_{i+1}})| < (1-\varepsilon)\hat{t}$, $|neg_{\Phi_i}(\beta_{Q_{i+1}}) \cap neg_{\Phi_i}(\beta_S)|$ (red shaded set) is also $< (1-\varepsilon)\hat{t}$. Thus, the remaining clauses in $neg_{\Phi_i}(\beta_S)$ are those that contain only variables from Q_{i+1} as negative literals and its size is strictly more than $\varepsilon\hat{t}$.

This completes the proof of the lemma. □

Due to Lemma 4, we get that if there is an optimum solution S such that $|\mathsf{pos}_\Phi(\beta_S)| \geq \frac{2k}{\varepsilon}\left(k + \frac{4kd}{\varepsilon}\right)^d d$, then either $S \cap X \neq \emptyset$, where X is the $\left(1 + \frac{1}{\varepsilon}\right)\left(k + \frac{4dk}{\varepsilon}\right)$ variables with the highest positive degrees, or X contains a good approximation. Thus, either we can find an approximate solution or "find" a vertex in S. To handle the other case, we design an algorithm using random separation. We use the notion of universal sets[1] for this.

We will also need the following result by Bläser [25].

Proposition 6 (Bläser [25])**.** *There is an algorithm that given a universe $\mathcal{U}$, a family of sets $\mathcal{F}$ over $\mathcal{U}$, and positive integers k and t, runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$, and*

[1]The reader can refer to the preliminaries in 4.1 for the definition.

check whether there exists a k sized subset of $\mathcal{U}$, that hits at least t sets from $\mathcal{F}$. If such a subset exists, then the algorithm will output such a subset.

Our next lemma describes an algorithm for cases when our general scheme does not work.

Lemma 5. *Let (Φ, k) be an instance of* CC-MaxSat*, and let $\varepsilon > 0$. Then there exists an algorithm that, given (Φ, k), $\varepsilon > 0$, and $t \in \mathbb{N}$, runs in time*

$$2^{\mathcal{O}(t/\varepsilon)} \cdot (n+m)^{\mathcal{O}(1)},$$

and outputs a solution $Y \subseteq \mathcal{V}_\Phi$ of size at most k such that the number of clauses satisfied by the corresponding assignment β_Y is at least $(1-\varepsilon) \cdot \mathsf{OPT}_k(\Phi)$, provided there exists an optimal solution S with $|\mathsf{pos}_\Phi(\beta_S)| = t$.

Proof. Let $\mathcal{V}_\Phi = \{v_1, v_2, \ldots, v_n\}$ denote the set of variables in Φ, and let $\mathcal{C}_\Phi = \{C_1, C_2, \ldots, C_m\}$ denote the set of clauses.

Our algorithm constructs a family $\mathcal{S}$ of candidate assignments (solutions) and ultimately outputs the one that satisfies the maximum number of clauses. It adds to $\mathcal{S}$ subsets of variables of size at most k, where each such subset $Z \in \mathcal{S}$ corresponds to an assignment β_Z. Recall that given a variable subset Z, the assignment β_Z is defined as follows: for each variable x, $\beta_Z(x) = 1$ if and only if $x \in Z$, and $\beta_Z(x) = 0$ otherwise. Initially, we set $\mathcal{S} := \emptyset$. Finally, the algorithm outputs a set $O_{\mathsf{max}} \in \mathcal{S}$ for which the corresponding assignment $\beta_{O_{\mathsf{max}}}$ satisfies the largest number of clauses among all assignments β_Z for $Z \in \mathcal{S}$.

Recall that S is an optimal solution such that the number of clauses satisfied positively by the assignment β_S is t, i.e., $|\mathsf{pos}_\Phi(\beta_S)| = t$. Our algorithm is divided into two parts.

Case I: $t \leq \varepsilon \cdot \mathsf{OPT}_k(\Phi)$. First, assume that $t \leq \varepsilon \cdot \mathsf{OPT}_k(\Phi)$. In this case, observe that the assignment which sets all variables to 0 satisfies all clauses that contain only negative literals. Since t represents the number of clauses that require at least one positive literal from the optimal solution, the number of clauses satisfied by setting all variables to 0 is at least $(1-\varepsilon) \cdot \mathsf{OPT}_k(\Phi)$.

To handle this case, we explicitly add the empty set $\emptyset$ to $\mathcal{S}$, corresponding to the assignment $\beta_\emptyset$ which sets all variables to $\{0\}$. As argued, $\beta_\emptyset$ satisfies at least $(1-$

$\varepsilon) \cdot \mathsf{OPT}_k(\Phi)$ clauses in Φ. Therefore, when the algorithm outputs the set $O_{\mathsf{max}} \in \mathcal{S}$, the corresponding assignment $\beta_{O_{\mathsf{max}}}$ satisfies at least $(1-\varepsilon) \cdot \mathsf{OPT}_k(\Phi)$ clauses in Φ.

Case II: $t \geq \varepsilon \cdot \mathsf{OPT}_k(\Phi)$. We now consider the case where $t \geq \varepsilon \cdot \mathsf{OPT}_k(\Phi)$, which implies that $\mathsf{OPT}_k(\Phi) \leq \frac{t}{\varepsilon}$. For this regime, we design an exact algorithm instead of an approximation algorithm.

Recall that S is an optimal solution. In this case, we construct a set $\mathsf{POS} \uplus \mathsf{NEG}$ consisting of at most t/ε variables such that, once the assignments to these variables are fixed accordingly, we are guaranteed to satisfy all $\mathsf{OPT}_k(\Phi)$ clauses.

To construct these sets, consider each clause C that is satisfied by the assignment β_S. For each such clause, arbitrarily choose a literal $z \in C$ that is set to 1 under β_S. If z is a positive literal, we add the corresponding variable to the set POS; otherwise (i.e., if z is a negative literal), we add the corresponding variable to the set NEG.

Observe that a variable assigned the value 1 can only satisfy clauses in which it appears positively. Therefore, the sets POS and NEG satisfy the following properties:

1. $\mathsf{POS} \cap \mathsf{NEG} = \emptyset$,
2. $|\mathsf{POS}| \leq k$,
3. $|\mathsf{POS} \uplus \mathsf{NEG}| \leq \frac{t}{\varepsilon}$.

Observe that if we assign the value 1 to all variables in POS and the value 0 to all variables in NEG, then, regardless of how the remaining variables are assigned, we are guaranteed to satisfy at least $\mathsf{OPT}_k(\Phi)$ clauses. There are two main challenges with this approach:

1. **Constructing the sets POS and NEG.** These sets are not known in advance. To overcome this, we utilize the concept of (n, ℓ)-universal sets (see Definition 4), which will help us systematically explore all relevant assignments to a bounded number of variables.
2. **Respecting the budget constraint on variables set to 1.** Assigning value 1 to all variables in POS may result in selecting more than k variables, violating our constraint. To handle this, we reduce the problem to an instance of the Hitting Set problem and apply Proposition 6.

Towards our algorithm, observe the following. If we can obtain a partition of the variable set $\mathcal{V}_\Phi = \mathsf{IncPOS} \uplus \mathsf{IncNEG}$ such that $\mathsf{POS} \subseteq \mathsf{IncPOS}$ and $\mathsf{NEG} \subseteq \mathsf{IncNEG}$, then we can proceed as follows.

1. We now set all variables in IncNEG to 0. This ensures that all clauses which were satisfied by assigning variables from NEG to 0 remain satisfied. Indeed, let $\mathcal{C}_{\mathsf{IncNEG}} \subseteq \mathcal{C}_\Phi$ denote the set of clauses that contain at least one variable from IncNEG appearing negatively. That is, every clause in $\mathcal{C}_{\mathsf{IncNEG}}$ is guaranteed to be satisfied under the assignment where all variables in IncNEG are set to 0.
2. Rather than setting all variables in IncPOS to 1, we proceed as follows:
   (a) Let $\mathcal{U} := \mathsf{IncPOS}$; that is, we consider all variables in the set IncPOS as candidates for being set to 1.
   (b) We construct a set family $\mathcal{F}_\mathcal{U}$ over the universe $\mathcal{U}$ as follows. Let $\mathcal{C}_{\mathsf{rem}} := \mathcal{C}_\Phi \setminus \mathcal{C}_{\mathsf{IncNEG}}$ be the set of clauses that are not already satisfied by setting variables in IncNEG to 0. For every clause $C \in \mathcal{C}_{\mathsf{rem}}$, define a set $Q_C \in \mathcal{F}_\mathcal{U}$, where Q_C consists of the variables that appear as positive literals in $C \setminus \mathsf{IncNEG}$.

We now use the set family $\mathcal{F}_\mathcal{U}$ to define a Hitting Set instance. Our goal is to select a subset $S_\mathcal{U} \subseteq \mathcal{U}$ of size at most k that intersects (or *hits*) as many sets in $\mathcal{F}_\mathcal{U}$ as possible. Each such hit corresponds to satisfying a clause in $\mathcal{C}_{\mathsf{rem}}$.

To achieve this, we invoke Proposition 6, which guarantees that for any integer $t_\mathcal{U} \leq |\mathcal{F}_\mathcal{U}|$, we can either find a set $S_\mathcal{U} \subseteq \mathcal{U}$ of size at most k that hits at least $t_\mathcal{U}$ sets in $\mathcal{F}_\mathcal{U}$, or correctly determine that no such set exists. We run this algorithm with $t_\mathcal{U} := \mathsf{OPT}_k(\Phi) - |\mathcal{C}_{\mathsf{IncNEG}}|$, and obtain a hitting set $S_\mathcal{U} \subseteq \mathcal{U}$ of size at most k that intersects at least $t_\mathcal{U}$ sets in $\mathcal{F}_\mathcal{U}$. We then add $S_\mathcal{U}$ to the solution family $\mathcal{S}$.

The corresponding assignment $\beta_{S_\mathcal{U}}$, which sets all variables in $S_\mathcal{U}$ to 1 and all remaining variables—including those in IncNEG—to 0, satisfies the following:

- All clauses in $\mathcal{C}_{\mathsf{IncNEG}}$, by setting the variables in IncNEG to 0,
- At least $t_\mathcal{U}$ clauses in $\mathcal{C}_{\mathsf{rem}}$, by construction of the hitting set.

Thus, the total number of satisfied clauses is at least $|\mathcal{C}_{\mathsf{IncNEG}}| + t_\mathcal{U}$. Since $\mathsf{POS} \subseteq \mathsf{IncPOS}$, and we know there exists a set of at most k variables (namely, the optimal

solution) that satisfies at least $\mathsf{OPT}_k(\Phi) - |\mathcal{C}_{\mathsf{IncNEG}}|$ clauses in $\mathcal{C}_{\mathsf{rem}}$, it follows that $t_{\mathcal{U}} \geq \mathsf{OPT}_k(\Phi) - |\mathcal{C}_{\mathsf{IncNEG}}|$. Putting this together, we conclude that the assignment $\beta_{S_{\mathcal{U}}}$ satisfies at least $\mathsf{OPT}_k(\Phi)$ clauses in total.

We summarize our findings as follows. If we are given a partition of the variable set $\mathcal{V}_\Phi = \mathsf{IncPOS} \uplus \mathsf{IncNEG}$ such that $\mathsf{POS} \subseteq \mathsf{IncPOS}$ and $\mathsf{NEG} \subseteq \mathsf{IncNEG}$, then in time

$$2^{t_{\mathcal{U}} := \mathsf{OPT}_k(\Phi) - |\mathcal{C}_{\mathsf{IncNEG}}|} \cdot (n+m)^{\mathcal{O}(1)},$$

we can compute a set $S_{\mathcal{U}} \subseteq \mathsf{IncPOS}$ of size at most k, along with the corresponding assignment $\beta_{S_{\mathcal{U}}}$, that satisfies at least $\mathsf{OPT}_k(\Phi)$ clauses in total. We refer to the above algorithm as `PartitionRespecting`.

Thus, our task reduces to finding a partition of the variable set $\mathcal{V}_\Phi = \mathsf{IncPOS} \uplus \mathsf{IncNEG}$ such that $\mathsf{POS} \subseteq \mathsf{IncPOS}$ and $\mathsf{NEG} \subseteq \mathsf{IncNEG}$.

A random partition of $\mathcal{V}_\Phi$ into two subsets A and B, where each variable is independently placed in A with probability $\frac{1}{2}$, yields such a partition with positive probability. Specifically, by taking $A = \mathsf{IncPOS}$ and $B = \mathsf{IncNEG}$, we obtain a valid partition with probability at least

$$\frac{1}{2^{|\mathsf{POS} \uplus \mathsf{NEG}|}} \geq \frac{1}{2^{t/\varepsilon}}.$$

We do not follow a randomized approach, and instead present a derandomized version using an $(n, \frac{t}{\varepsilon})$-universal set.

1. Invoke the algorithm from Proposition 2 to obtain an $(n, \frac{t}{\varepsilon})$-universal set $\mathcal{U}$.
2. For each $Z \in \mathcal{U}$, construct a partition $A_Z \uplus B_Z$ of the variable set $\mathcal{V}_\Phi$ as follows: let $A_Z := \{v_i \mid i \in Z\}$ and $B_Z := \mathcal{V}_\Phi \setminus A_Z$. Run the `PartitionRespecting` algorithm with input (A_Z, B_Z), and add the resulting set to the solution family $\mathcal{S}$.
3. Finally, return the set $O_{\mathsf{max}} \in \mathcal{S}$ that maximizes the number of satisfied clauses under the corresponding assignment $\beta_{O_{\mathsf{max}}}$.

We claim that there exists a partition among those generated from the universal set such that $\mathsf{POS} \subseteq \mathsf{IncPOS}$ and $\mathsf{NEG} \subseteq \mathsf{IncNEG}$. By the property of an $(n, \frac{t}{\varepsilon})$-

universal set, we know that for every subset $S \subseteq [n]$ of size at most $\frac{t}{\varepsilon}$, the family $\{W \cap S : W \in \mathcal{U}\}$ contains all $2^{|S|}$ subsets of S.

Let S be the set of indices corresponding to the variables in $\mathsf{POS} \uplus \mathsf{NEG}$; that is, $i \in S$ if and only if $v_i \in \mathsf{POS} \cup \mathsf{NEG}$. By the universality property, there exists a set $Z \in \mathcal{U}$ such that $Z \cap S$ contains exactly the indices corresponding to POS, and none of those corresponding to NEG. Therefore, the partition of $\mathcal{V}_\Phi$ induced by Z, with $A_Z := \{v_i \mid i \in Z\}$ and $B_Z := \mathcal{V}_\Phi \setminus A_Z$, satisfies the required conditions: $\mathsf{POS} \subseteq A_Z = \mathsf{IncPOS}$ and $\mathsf{NEG} \subseteq B_Z = \mathsf{IncNEG}$. This completes the proof of correctness of the algorithm.

Running Time. The running time of the algorithm is bounded by the product of the size of the $(n, \frac{t}{\varepsilon})$-universal set and the running time of the `PartitionRespecting` algorithm. By Proposition 2, the size of the universal set is at most

$$2^{\frac{t}{\varepsilon}} \left(\frac{t}{\varepsilon}\right)^{\mathcal{O}(\log(t/\varepsilon))} n \log n,$$

and the running time of `PartitionRespecting` is $2^{\mathcal{O}(t/\varepsilon)}(n+m)^{\mathcal{O}(1)}$.

Therefore, the total running time of the algorithm is

$$2^{\mathcal{O}(t/\varepsilon)}(n+m)^{\mathcal{O}(1)}.$$

This completes the proof. □

Now we combine Lemmata 4 and 5 and prove Theorem 1, which has been stated below.

Theorem 1. *There is an algorithm that given an instance (Φ, k) of* CC-MaxSat *and $\epsilon > 0$ such that G_Φ is $K_{d,d}$-free, runs in time $2^{\mathcal{O}((\frac{dk}{\epsilon})^d)}(n+m)^{\mathcal{O}(1)}$, and outputs a solution Y such that the number of clauses satisfied by β_Y is at least $(1-\epsilon)\mathsf{OPT}_k(\Phi)$.*

Proof. Let $q = k + \frac{4dk}{\varepsilon}$ and $q^\star = \left(1 + \frac{1}{\varepsilon}\right) q$. Let $t^\star = \frac{2k}{\varepsilon} q^d d$. Let $X = \{v_1, \ldots, v_{q^\star}\}$ be the first $q^\star$ variables with the highest positive degrees. We output the best solution among the set of solutions $\mathcal{S}$ constructed below.

Let S be an optimum solution. Now if $|\mathsf{pos}_\Phi(\beta_S)| \geq t^\star$, then we have $\frac{\varepsilon t^\star}{2k} \geq q^d d$. Thus, by Lemma 4, we know that either $S \cap X \neq \emptyset$ or there is a solution $Y \subseteq X$ such that the number of clauses satisfied by β_Y is at least $(1-\varepsilon)\mathsf{OPT}_k(\Phi)$. On the other hand, if $|\mathsf{pos}_\Phi(\beta_S)| < t^\star$, then we can use Lemma 5.

Initially $\mathcal{S} := \{Y \subseteq X : |Y| \leq k\}$.

1. For each $v \in X$, let Φ_v be the formula obtained from Φ as follows. Delete all the clauses that contain the literal v. Delete the literal $\overline{v}$ from the clauses that contains it. Recursively, solve $(\Phi_v, k-1)$ and let S_v be the solution. Then, we add $S_v \cup \{v\}$ to $\mathcal{S}$.

2. For each $t \in [t^\star]$, run the algorithm in Lemma 5 and add the solution to $\mathcal{S}$.

The correctness of the algorithm follows from Lemmata 4 and 5. For the running time of the algorithm, we get the following recurrence relation.

$$T(n, m, k) = q^\star T(n, m, k-1) + 2^{\mathcal{O}(\frac{2k}{\varepsilon^2} q^d d)}(n+m)^{\mathcal{O}(1)} \tag{5.4}$$

Here, the base case is $T(n, m, 0) = 1$. Then, $T(n, m, k)$ is upper bounded by $(q^\star)^k 2^{\mathcal{O}(\frac{2k}{\varepsilon^2} q^d d)}(n+m)^{\mathcal{O}(1)}$. By substituting the values of q and $q^\star$, $T(n, m, k)$ is upper bounded by $2^{\mathcal{O}((\frac{dk}{\varepsilon})^d)}(n+m)^{\mathcal{O}(1)}$. This completes the proof of the theorem. □

5.2 EPAS for Max Coverage on $K_{d,d}$-free instances

In this chapter, we design an EPAS parameterized by the solution size k for Maximum Coverage on $K_{d,d}$-free formulas.

Notice that the algorithm given in Theorem 1 also implies an algorithm for Maximum Coverage. However, using the fact that it does not have any negative literals, we can design a much faster algorithm for it. With the help of incidence graph, we first recast Maximum Coverage to Max Red Blue Dominating Set.

Max Red Blue Dominating Set (MaxRBDS) **Parameter:** k
Input: A bipartite graph G, with $V(G) = A \uplus B$, and an integer k.
Output: A set $S \subseteq A$ of size at most k such that $|N(S)| = \mathsf{OPT}_k(G)$.

Recall that $\mathsf{OPT}_k(G)$ is the maximum size of a neighborhood of any set of k vertices from A.

For our approximation algorithm, we also need an algorithm for the decision version of the MaxRBDS, namely, Dec MaxRBDS. Using Proposition 6, we get the following result.

Theorem 2 (Bläser [25]). Dec MaxRBDS *can be solved in time* $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.

Using the above proposition, we get the following result.

Theorem 3. *There exists an algorithm for* MaxRBDS*, that given a* $K_{d,d}$*-free bipartite graph* G*, with* $V(G) = A \uplus B$*, an integer* k*, and an* $\varepsilon > 0$*, runs in time* $\left(\frac{dk}{\varepsilon}\right)^{\mathcal{O}(dk)} n^{\mathcal{O}(1)}$*, and returns a set* $S \subseteq A$ *of size* k *such that* $|N(S)| \geq (1-\varepsilon)\mathsf{OPT}_k(G)$.

Our starting point for this EPAS is also the state-of-the-art EPAS for Max k-Vertex Cover and its generalization [19]. The key idea to design the algorithm is the following. Let x be a highest degree vertex in A, and H be the subset of A such that for all $y \in H$, we have that $|N(y) \cap N(x)| \geq \frac{\varepsilon|N(x)|}{k}$. That is, H consists of vertices in A that have "high" intersection with the neighbors of x. Then, one can show that there exists a set $S^\star \subseteq A$ such that $|S^\star| \leq k$, $S^\star \cap H \neq \emptyset$ and $|N(S^\star)| \geq (1-\varepsilon)\mathsf{OPT}_k(G)$.

Our main idea immediately leads to a recursive algorithm by branching on vertices in $H \cup \{x\}$. However, to bound the number of recursive calls, we first bound the size of H as a function of k, d, ε. This combinatorial lemma (Lemma 6) could be of independent interest, and indeed we *crucially use this* to prove our result. However, the bound on H only holds when the degree of x is at least $\frac{kd}{\varepsilon}$. Fortunately, in the case when the degree of x is at most $\frac{kd}{\varepsilon}$, we have that $\mathsf{OPT}_k(G) \leq \frac{k^2 d}{\varepsilon}$. Thus, in this case, we can apply an exact algorithm by Bläser [25] and obtain a set S of size k such that $|N(S)| = \mathsf{OPT}_k(G)$. This algorithm runs in time $2^{\mathcal{O}(k^2 d/\varepsilon)}n^{\mathcal{O}(1)}$. To improve the running time of this algorithm, we take a detour and design a polynomial kernel for the decision version of MaxRBDS, which is used to obtain the claimed running time of the EPAS for MaxRBDS. The polynomial kernel is of size $\left(\frac{kd}{\varepsilon}\right)^{\mathcal{O}(dk)}$. Trying all k sized subsets of the vertex set of G' gives us our result. We present the kernel in Section 9.3.

For the proof of Theorem 3, we apply the following strategy: we show that there exists a $(1-\varepsilon)$-approximate solution that either contains a highest degree vertex $x \in A$, or at least one vertex from the set $Q \subseteq A$ of vertices where for each vertex $y \in Q$, $|N(x) \cap N(y)|$ is *large*. Moreover, we will prove that the cardinality of Q is *small*. We first define what we mean by vertices that have large neighborhood in B.

Definition 10 (β-High Degree Set). *Given a bipartite graph* $G = (A, B, E)$*, a subset* $B' \subseteq B$ *and a positive integer* $\beta > 1$*, the* β-High Degree Set *of* B'*, denoted by* $\mathsf{HD}^G_\beta(B') \subseteq A$*, is a set of vertices such that every vertex* $v \in \mathsf{HD}^G_\beta(B')$ *satisfies* $|N(v)| \geq \frac{|B'|}{\beta}$*, i.e.,*

$$\mathsf{HD}^G_\beta(B') = \{w : w \in A, |N(w) \cap B'| \geq \tfrac{|B'|}{\beta}\}$$

In general, in any bipartite graph the size of $\mathsf{HD}^G_\beta(B)$ may be unbounded. However, the next lemma shows that in $K_{d,d}$-free graphs, it is "almost" bounded by an appropriate function of β, d. Let

$$\mathsf{AHD}^G_\beta(B') = \mathsf{HD}^G_\beta(B') \cap \{v \mid v \in A, |N(v)| \geq d\}. \tag{5.5}$$

That is, $\mathsf{AHD}^G_\beta(B')$ consists of vertices of degree at least d and those that belong to $\mathsf{HD}^G_\beta(B')$.

Lemma 6. *For all d and for all $\beta > 1$ with $\frac{|B|}{2\beta} > d$, if $G = (A, B, E)$ is a $K_{d,d}$-free bipartite graph, then $|\mathsf{AHD}^G_\beta(B)| \leq f(\beta, d) = (d-1)(2\beta)^d$.*

Proof. Let K the set containing all induced subgraphs in G that are isomorphic to $K_{1,d}$, such that the part with *single* vertex comes from A and the part with d *sized subset* comes from B. We argue that since G is $K_{d,d}$-free, $|K| \leq (d-1)\binom{|B|}{d}$. Indeed, if $|K| > (d-1)\binom{|B|}{d}$, then there exists a set in B of size d with d common neighbors in A, contradicting that G is $K_{d,d}$-free. Note that, every vertex v in the set $\mathsf{AHD}^G_\beta(B)$ has at least $\frac{|B|}{\beta}$ neighbors in B, and thus any set of size d in $N(v)$ together with v contributed to a $K_{1,d}$ in the set K. Thus, every vertex in the set $\mathsf{AHD}^G_\beta(B)$ contributes to at least $\binom{\frac{|B|}{\beta}}{d}$ many $K_{1,d}$ in the set K. Thus the number of vertices in $\mathsf{AHD}^G_\beta(B)$ is upper bounded by

$$\frac{(d-1)\binom{|B|}{d}}{\binom{\frac{|B|}{\beta}}{d}} \leq \frac{(d-1)(|B|)!}{(|B|-d)!d!} \cdot \frac{(\frac{|B|}{\beta}-d)!d!}{(\frac{|B|}{\beta})!} \leq \frac{(d-1)(|B|)^d}{(\frac{|B|}{\beta}-d)^d} < \frac{(d-1)(|B|)^d}{(\frac{|B|}{\beta}-\frac{|B|}{2\beta})^d}$$

$$\leq \frac{(d-1)(|B|)^d}{(\frac{|B|}{2\beta})^d} \leq (d-1)(2\beta)^d.$$

This completes the proof. □

An alternate proof could also be obtained using the bound on the Zarankiewicz function, $z(m, n; s, t)$, proved by Hyltén-Cavallius [70] using the Kővári–Sós–Turán theorem [71]. We provide a brief sketch of the argument below.

Given a bipartite graph $G = (A \uplus B, E)$ where $|A| = m$ and $|B| = n$, and assuming the graph does not contain the complete bipartite graph $K_{s,t}$ as a subgraph, the

function $z(m, n; s, t)$ denotes the maximum possible number of edges in G. The bound provided by Hyltén-Cavallius is as follows

$$z(m, n; s, t) < (s-1)^{1/t}(n-t+1)m^{1-1/t} + (t-1)m.$$

Consider the subgraph induced on the vertices in $\mathsf{AHD}^G_\beta(B)$ and B. Let $m = |\mathsf{AHD}^G_\beta(B)|$. Since the graph is $K_{d,d}$-free we have

$$z(m, |B|; d, d) < (d-1)^{1/d}(|B|-d+1)m^{1-1/d} + (d-1)m.$$

Since each vertex in $\mathsf{AHD}^G_\beta(B)$ has at least $\frac{|B|}{\beta}$ neighbors in this subgraph, a lower bound on $z(m, |B|; d, d)$ is $m\frac{|B|}{\beta}$. Thus, we get:

$$\begin{aligned}
m \cdot \frac{|B|}{\beta} &< (d-1)^{1/d}(|B|-d+1)\, m^{1-1/d} + (d-1)m \\
\implies m\left(\frac{|B|}{\beta} - (d-1)\right) &< (d-1)^{1/d}(|B|-d+1)\, m^{1-1/d} \\
\implies m^{1/d} &< (d-1)^{1/d} \cdot \frac{|B|-d+1}{\frac{|B|}{\beta} - (d-1)} \\
\text{Assuming } \frac{|B|}{\beta} > 2d, \text{ we have } \frac{|B|-d+1}{\frac{|B|}{\beta} - (d-1)} &< 2\beta \\
\implies m^{1/d} &< (d-1)^{1/d} \cdot 2\beta \\
\implies m &< (d-1)(2\beta)^d
\end{aligned}$$

Lemma 6 provides the required upper bound on the number of vertices that can have high intersection with neighbors of a particular vertex. Using this combinatorial upper bound and an exact algorithm for Dec MaxRBDS, we give an EPAS for MaxRBDS.

Proof of Theorem 3. An input to our problem consists of a $K_{d,d}$-free bipartite graph $G = (A, B, E)$, an integer k and $\varepsilon \in (0, 1)$. We give a recursive algorithm, APX-RBDS, that selects a vertex of highest degree x and recursively searches for a good approximate solution by deciding to include one of the vertices in $\mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$, for an appropriately selected β. We refer to Algorithm 1 for a detailed description of the algorithm.

Approximation Ratio. We show that the algorithm returns a set $S \subseteq A$, of size k, such that $|N(S)| \geq (1-\varepsilon)\mathsf{OPT}_k(G)$ by induction on k, the size of the solution. We

Algorithm 1 APX-RBDS$(G = (A, B, E), k, \varepsilon)$

if $k \leq 0$, **then**
  return $\emptyset$.
end if
if $\Delta(A) \leq 2\beta d$, **then**
  Apply Theorem 2 and obtain S of size k such that $|N(S)| = \mathsf{OPT}_k(G)$, where $\beta = \frac{k}{\varepsilon}$.
  return S.
else
  Let $x \in A$, such that $\mathsf{deg}(x) = \Delta(A)$.
  Compute $\mathsf{AHD}^G_\beta(N(x))$ for $\beta = \frac{k}{\varepsilon}$.
  for each $y \in \mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$, **do**
    Let $G_y = G[A_y, B \setminus N(y)]$ and $A_y = A \setminus \{y\}$.
    Let S_y be the set returned by APX-RBDS$(G_y, k-1, \varepsilon)$.
  end for
  Among all the sets S_y, $y \in \mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$, let z be the element such that $N(\{z\} \cup S_z)$ is maximized. **return** $\{z\} \cup S_z$.
end if

set $\beta = \frac{k}{\varepsilon}$. Let $\Delta(A)$ denote the highest degree of a vertex in the set A.

Base Case. When $k = 0$, we have that $\mathsf{OPT}_0(G) = 0$, and thus the statement holds.

Inductive Case. Here, we have two cases based on whether $\Delta(A) \leq 2\beta d$ or not. In the first case, $\mathsf{OPT}_k(G) \leq k \cdot \Delta(A) \leq 2k\beta d$. Thus, we have that $\mathsf{OPT}_k(G)$ is bounded by a function of k, d and ε, and hence by applying Theorem 2, we can *exactly* solve the problem. More specifically, we apply Theorem 2 on $(G = (A, B, E), k, t)$, by increasing t from $\Delta(A)$ to $k\Delta(A)$, and find a set $S \subseteq A$, of size k, such that $|N(S)| = \mathsf{OPT}_k(G)$. Correctness of this step follows from the correctness of Theorem 2.

Finally, we reach a case where $\Delta(A) > 2\beta d$. Let $x \in A$, such that $\mathsf{deg}(x) = \Delta(A)$. Further, compute $\mathsf{AHD}^G_\beta(N(x))$. The algorithm for each $y \in \mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$, does as follows. Let $G_y = G[A_y, B \setminus N(y)]$, where $A_y = A \setminus \{y\}$. The algorithm recursively solves the problem by calling APX-RBDS$(G_y, k-1, \varepsilon)$ for $y \in \mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$. In this case we will show that there exists a $z \in \mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$ such that $|N(\{z\} \cup S_z)| \geq (1-\varepsilon)\mathsf{OPT}_k(G)$ (where S_z is the set returned by APX-RBDS$(G_z, k-1, \varepsilon)$). Let $Z := \mathsf{AHD}^G_\beta(N(x)) \cup \{x\}$.

Let $O \subseteq A$ be a set of size k, such that $|N(O)| = \mathsf{OPT}_k(G)$. That is, O is an optimum solution. For our proof we distinguish two cases based on whether $O \cap Z \neq \emptyset$ or $O \cap Z = \emptyset$

- **Case 1** ($O \cap Z \neq \emptyset$) : Let $z \in (O \cap Z)$ and let $O' = O \setminus \{z\}$. Observe that $|N_{G_z}(O')| = \mathsf{OPT}_{k-1}(G_z)$, otherwise it will contradict that O is an optimum solution. Thus, we have that $\mathsf{OPT}_k(G) = |N(z)| + \mathsf{OPT}_{k-1}(G_z)$. By induction hypothesis, we have that APX-RBDS, when called on $(G_z, k-1, \varepsilon)$, returns a set S_z such that $|N_{G_z}(S_z)| \geq (1-\varepsilon)\mathsf{OPT}_{k-1}(G_z)$. Thus, the size of $N(\{z\} \cup S_z)$ is given by

$$\begin{aligned} |N(\{z\} \cup S_z)| &= |N(z)| + |N_{G_z}(S_z)| \\ &\geq |N(z)| + (1-\varepsilon)\mathsf{OPT}_{k-1}(G_z) \\ &\geq (1-\varepsilon)(|N(z)| + \mathsf{OPT}_{k-1}(G_z)) = (1-\varepsilon)(\mathsf{OPT}_k(G)) \end{aligned}$$

- **Case 2** ($O \cap Z = \emptyset$) : In this case, we argue that the branch that includes the element x will give the desired set S. Let f be a vertex in O such that $|N(f) \setminus N(O \setminus \{f\})|$ is minimized among all vertices in O. That is the number of vertices dominated by f alone in the solution O is minimized. Then, $O' = O \setminus \{f\}$ has at least $\mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k}$ neighbors in B. That is, $|N_G(O')| \geq \mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k}$. Furthermore, since we are in the case that $\Delta(A) > 2\beta d$, we have that $\frac{|N(x)|}{2\beta} \geq d$. This implies that if $w \notin \mathsf{AHD}^G_\beta(N(x))$, then $|N(w) \cap N(x)| < \frac{|N(x)|}{\beta}$. Thus, we get that

$$\begin{aligned} \mathsf{OPT}_{k-1}(G_x) &\geq |N(O') \setminus N(x)| \\ &\geq \mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k} - k\frac{|N(x)|}{\beta} \\ &\geq \mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k} - \varepsilon|N(x)|. \end{aligned}$$

Having armed ourselves with a lower bound on $\mathsf{OPT}_{k-1}(G_x)$, we proceed to show the desired approximation ratio for the set $S = S_x \cup \{x\}$.

$$\begin{aligned} |N(\{x\} \cup S_x)| &= |N(x)| + |N_{G_x}(S_x)| \\ &\geq |N(x)| + (1-\varepsilon)\mathsf{OPT}_{k-1}(G_x) \\ &\geq |N(x)| + (1-\varepsilon)\left(\mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k} - \varepsilon|N(x)|\right) \\ &\geq (1-\varepsilon)|N(x)| + (1-\varepsilon)\left(\mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k}\right) \\ &\geq (1-\varepsilon)\left(\mathsf{OPT}_k(G) - \frac{\mathsf{OPT}_k(G)}{k} + |N(x)|\right) \geq (1-\varepsilon)\mathsf{OPT}_k(G). \end{aligned}$$

The last inequality follows from the fact that $|N(x)| \geq \frac{\mathsf{OPT}_k(G)}{k}$.

Running Time. The running time of the algorithm is governed by either the recurrence

$$T(k,\varepsilon) \leq |\mathsf{AHD}^G_\beta(N(x)) \cup \{x\}| \cdot T(k-1,\varepsilon)$$

or it is dominated by the running time of algorithm given by Theorem 2. In the former case, the running time is upper bounded by

$$\left((d-1)\left(\frac{2k}{\varepsilon}\right)^d\right)^k n^{\mathcal{O}(1)} = \left(\frac{dk}{\varepsilon}\right)^{\mathcal{O}(dk)} n^{\mathcal{O}(1)}.$$

In the latter case, the algorithm runs in time $2^{\mathcal{O}\left(\frac{2k^2d}{\varepsilon}\right)} n^{\mathcal{O}(1)}$ by invoking Theorem 2. However, this running time of the described algorithm does not match with the claimed running time. Thus, in order to improve the running time of the whole algorithm, we focus on designing faster algorithm for the problem when $\Delta(A) \leq 2\beta d$. In this case, $\mathsf{OPT}_k(G) \leq k \cdot 2\beta d = 2d \cdot \frac{k^2}{\varepsilon}$. In Chapter 9.3, we design a kernel for the problem with $\mathcal{O}(t^{d+1}d)$ vertices, where $t = \mathsf{OPT}_k(G)$. Thus, when $\Delta(A) \leq 2\beta d$, we apply this kernelization for all $t \in \{\Delta(A), \Delta(A)+1, \ldots, 2d \cdot \lceil \frac{k^2}{\varepsilon} \rceil\}$ and obtain a solution by brute force search. This running time is upper bounded by $\left(\frac{dk}{\varepsilon}\right)^{\mathcal{O}(dk)} n^{\mathcal{O}(1)}$. This completes the proof of the theorem. □

Chapter 6

From Max k-weight Satisfiability to Partial Coverage: a randomized reduction

We present a reduction in this chapter that establishes the equivalence of CC-MaxSat and Maximum Coverage in the context on FPT approximation parameterized by the solution size. This reduction works in the presence of additional constraints such as fairness constraints on the clauses (or elements) and matroid constraints on the variables (or sets) and preserves the structural properties of the input instance as long as it belongs to a subgraph-closed graph class. The success probability of the reduction is $\mathcal{O}((\varepsilon/r)^k)$, where r is the number of fairness constraints defined on the input instance. Hence, any result for a variant of Maximum Coverage carries forward to the same variant of CC-MaxSat with constant probability with a multiplicative factor of $\mathcal{O}((\varepsilon/r)^k)$ in the running time. We specifically give a polynomial-time approximate-preserving randomized reduction from (M, F)-Max k-weight SAT to (M, F)-Maximum Coverage to illustrate its applicability for the most general variant of the problem.

6.1 An overview of the reduction from CC-MaxSat to Maximum Coverage

This theorem is essentially a randomized *approximation-preserving reduction* from CC-MaxSat to Maximum Coverage. Given an instance $\mathcal{I} = (\Phi, k)$ of CC-

MaxSat, we first compute a random assignment Ψ that assigns a variable independently to be 1 with probability $p = \varepsilon/2$ and 0 with probability $1 - p$. Let V^* be the set of at most k variables set to be 1 by an optimal assignment Ψ^*. It is straightforward to see that, the probability that all the variables in V^* are set to be 1 by the random assignment Ψ is p^k – we say that this is the good event $\mathcal{G}$. Now, consider a clause that is satisfied negatively by Ψ^*, i.e., a clause C that contains a negative literal $\neg x$ and $\Psi^*(x) = 0$. It is also easy to see that, conditioned on the good event $\mathcal{G}$, the probability that such a clause C is also satisfied negatively by Ψ is at least $1 - p$. Thus, the expected number of clauses that are satisfied negatively by Ψ, conditioned on $\mathcal{G}$, is at least $1 - p$ times the number of clauses satisfied negatively by Ψ^*. Markov's inequality implies that, with probability at least $1/2$, the actual number of such clauses is close to its expected value. Thus, conditioned on $\mathcal{G}$, and the previous event, we can focus on the positively satisfied clauses (note that the probability that both of these events occur is at least $1/2 \cdot (\varepsilon/2)^k$. To this end, we can eliminate all the negatively satisfied clauses, and we can also prune the remaining clauses by eliminating any negative literals and the variables that are set to 0 by Ψ. Thus, all the remaining clauses only contain positive literals, which can be seen as an instance $\mathcal{I}'$ of Maximum Coverage. Furthermore, conditioned on $\mathcal{G}$, the variables set to 1 by Φ^* correspond to a family $\mathcal{F}^*$ of size k, and the elements covered by $\mathcal{F}^*$ correspond to the set of clauses satisfied only positively by Φ^*. Thus, if we find a $(1 - \varepsilon)$-approximate solution to $\mathcal{I}'$, and set the corresponding variables to 1, and the rest of the variables to 0, then we get a weight-k assignment that satisfies at least $(1 - \varepsilon) \cdot \mathsf{OPT}_{\mathsf{k}}^{\mathrm{SAT}}$ clauses, where $\mathsf{OPT}_{\mathsf{k}}^{\mathrm{SAT}}$ is the maximum number of clauses that can be satisfied by any assignment of weight at most k.

6.2 Randomized Reduction from Max k-weight SAT to Maximum Coverage

In the (M, F)-Max k-weight SAT problem, given a CNF-formula Φ with $\chi : \mathcal{C}_\Phi \to [r]$, a coverage demand function $t : [r] \to \mathbb{N}$, a matroid $\mathcal{M} = (\mathcal{V}, I)$, and an integer k, the goal is to find an assignment of weight at most k that satisfies at least $t(i)$ (also denoted as t_i) clauses of color class i, and the variables set to 1 by the assignment form an independent set in $\mathcal{M}$. An assignment Ψ satisfying these properties is called an *optimal weight-k assignment*.

We begin by recalling some notations. Let Φ be a CNF-formula. By $\mathcal{V}_\Phi$ and $\mathcal{C}_\Phi$, we

Algorithm 2 Reduction Algorithm($\mathcal{I} = (\Phi, r, \chi, \mathcal{M} = (\mathcal{V}_\Phi, I), k, t)$ of (M, F)-Max k-weight SAT)

1: Construct a random assignment Ψ as follows. For each variable $x \in \mathcal{V}_\Phi$, independently set $\Psi(x)$ to 1 with probability p and 0 with probability $1-p$. ▷ We will later set $p = \frac{\varepsilon}{2r}$.
2: Construct a new formula Φ' as follows:
- Let $N \subseteq \mathcal{C}_\Phi$ be the set of clauses that are satisfied negatively by Ψ. Then, $\mathcal{C}_{\Phi'} = \mathcal{C}_\Phi \setminus N$.
- For each $c \in \mathcal{C}_{\Phi'}$, remove all the variables in c that occur either as a negative literal or set to 0 by Ψ.
- For each $c \in \mathcal{C}_{\Phi'}$, add $\mathsf{var}(c)$ to $\mathcal{V}_{\Phi'}$.

3: Construct an instance $\mathcal{J}_\Psi = ((\mathcal{U}, \mathcal{F}), r', \chi', \mathcal{M}', k', t')$ of (M, F)-Maximum Coverage as follows:
- Set $\mathcal{U} = \mathcal{C}_{\Phi'}$.
- For each $v \in \mathcal{V}_{\Phi'}$, add a set f_v to $\mathcal{F}$ where $f_v = \{c \in \mathcal{C}_{\Phi'} : v \in \mathsf{var}(c)\}$.
- Define $\mathcal{M}' = (\mathcal{F}, I')$. For each $X \subseteq \mathcal{V}$, if $X \in I$, then $f_X \in I'$. Here f_X is the subfamily in $\mathcal{F}$ corresponding to X.
- For each $c \in \mathcal{C}_{\Phi'}$, if the corresponding element in $\mathcal{U}$ is e_c, set $\chi'(e_c) = \chi(c)$.
- Set $r' = r$, and $k' = k$.
- Set $t'(j) = t(j) - \frac{|N \cap \chi^{-1}(j)|}{(1-\varepsilon)}$ for each $j \in [r]$.

4: Return the instance $\mathcal{J}_\Psi$.

denote the set of variables and clauses in the formula Φ, respectively. An assignment to a CNF-SAT-formula Φ is a function $\Psi : \mathcal{V}_\Phi \to \{0, 1\}$. The weight of an assignment is the number of variables that have been assigned 1. By $T(\Psi)$ and $F(\Psi)$, we denote the set of variables assigned 1 and 0 by the assignment Ψ, respectively. For a clause $c \in \mathcal{C}_\Phi$, $\mathsf{var}(c)$ is the set of variables that occur in the clause c as a positive or negative literal. Similarly, for a set of clauses $C \in \mathcal{C}_\Phi$, $\mathsf{var}(C)$ is the set of variables that occur as a positive or negative literal in a clause $c \in C$.

Our reduction (Algorithm 2) takes an instance $(\Phi, r, \chi, \mathcal{M}, k, t)$ of (M, F)-Max k-weight SAT as input. It constructs a random assignment Ψ by setting each variable to 1 with probability p and 0 with probability $1-p$. It then constructs a new formula by first removing the set of clauses that are satisfied negatively by Ψ, followed by removing negative literals as well variables set to 0 by Ψ from the remaining clauses. It reduces the formula to an instance $((\mathcal{U}, \mathcal{F}), r', \chi', \mathcal{M}', k', t')$ of (M, F)-Maximum Coverage as described in Step 3 of Algorithm 2. Clearly, this reduction takes polynomial time.

Next, we prove the correctness of our reduction. For a yes instance $\mathcal{I}$ of (M, F)-Max k-weight SAT, let $\Psi^\star$ be an optimal assignment of weight at most k. Let $N^\star$ be the set of clauses satisfied negatively by $\Psi^\star$, i.e., every clause in $N^\star$ contains a negative literal that is set to 1, and let $P^\star$ be the set of clauses satisfied only positively by $\Psi^\star$, i.e., every clause in $P^\star$ contains a positive literal that is set to 1 and no negative literal in this clause is set to 1. By $N_i^\star$, we mean the set of clauses in color class i satisfied negatively by $\Psi^\star$ and by $P_i^\star$, we mean the set of clauses in color class i satisfied only positively by $\Psi^\star$. We call a random assignment, constructed in Algorithm 2, *good* if each variable in $T(\Psi^\star)$ (positive variables under $\Psi^\star$) is assigned 1 by Ψ, i.e., $T(\Psi^\star) \subseteq T(\Psi)$. A random assignment is good with probability at least p^k. For a good assignment Ψ, let N_i denote the set of clauses in color class i satisfied negatively by Ψ and P_i denote the set of clauses in color class i satisfied only positively by Ψ. We say that an event $\mathcal{G}$ is *good* if a good assignment Ψ is generated in Algorithm 2. We begin with the following claim.

Claim 3. *Given a yes instance $\mathcal{I}$ of* (M, F)-Max k-weight SAT*, with probability at least $1/2$, a good assignment Ψ satisfies at least $(1-\varepsilon)|N_i^\star|$ clauses negatively, for each $i \in [r]$.*

Proof. Let $(\Phi, r, \chi, \mathcal{M}, k, t)$ be a yes instance of (M, F)-Max k-weight SAT. Let Ψ be a good assignment, which occurs with probability at least p^k. We show that Ψ satisfies at least $(1-\varepsilon)|N^\star|$ clauses negatively, with probability at least $1/2$. Let X_i be the number of clauses in $N_i^\star$ that are satisfied negatively by Ψ. We define an indicator random variable x_j, for each $j \in [|N_i^\star|]$, as follows.

$$x_j = \begin{cases} 1 & \text{clause } c_j \in N_i^\star \text{ is satisfied negatively by } \Psi \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{Pr}(x_j \mid \mathcal{G}) = \mathsf{Pr}(\text{clause } c_j \in N_i^\star \text{ is satisfied negatively by } \Psi \mid \mathcal{G}) \geq (1-p)$$

$$\mathbb{E}[X_i \mid \mathcal{G}] = \sum_{j \in [|N_i^\star|]} x_j \times \mathsf{Pr}(x_j \mid \mathcal{G}) \geq (1-p)|N_i^\star|$$

Let $Y_i = |N_i^\star| - |N_i|$, where N_i is the set of clauses satisfied negatively by Ψ. Note that,

$$Y_i = |N_i^\star| - |N_i| \leq |N_i^\star \setminus N_i| = |N_i^\star| - X_i$$

Thus,

$$\mathbb{E}[Y_i \mid \mathcal{G}] \leq |N_i^\star| - \mathbb{E}[X_i \mid \mathcal{G}] \leq p|N_i^\star|$$

Since $Y_i \geq 0$, we can use Markov's inequality and get

$$\mathsf{Pr}(Y_i \geq 2rp|N_i^\star| \mid \mathcal{G}) \leq \frac{\mathbb{E}[Y_i]}{2rp|N_i^\star|} \leq \frac{1}{2r}$$

Since $Y_i = |N_i^\star| - |N_i|$, we get

$$\mathsf{Pr}(|N_i| \leq (1-2rp)|N_i^\star| \mid \mathcal{G}) \leq \frac{1}{2r}$$

By union bound,

$$\mathsf{Pr}\left(\exists i \in [r], |N_i| \leq (1-2rp)|N_i^\star| \mid \mathcal{G}\right) \leq \sum_{i\in[r]} \mathsf{Pr}(|N_i| \leq (1-2rp)|N_i^\star| \mid \mathcal{G}) \leq \frac{1}{2}$$

This implies that

$$\mathsf{Pr}\left(\forall i \in [r], |N_i| > (1-2rp)|N_i^\star| \mid \mathcal{G}\right) \geq \frac{1}{2}$$

Setting $p = \frac{\varepsilon}{2r}$ gives us the required result, i.e., with probability at least $1/2$, for all colors $i \in [r]$, $|N_i| > (1-\varepsilon)|N_i^\star|$. □

Lemma 7. *If $\mathcal{I} = (\Phi, r, \chi, \mathcal{M}, k, t)$ is a yes-instance of* (M, F)-Max k-weight SAT*, then with probability at least $(\frac{\varepsilon}{2r})^k$, the reduced instance $\mathcal{J}_\Psi = ((\mathcal{U}, \mathcal{F}), r', \chi', \mathcal{M}', k', t')$ is a yes-instance of* (M, F)-Maximum Coverage.

Proof. Let $\mathcal{I}$ be a yes instance of (M, F)-Max k-weight SAT and let $\Psi^\star$ be an optimal weight k assignment. Further, let $N^\star$ be the set of clauses satisfied negatively by $\Psi^\star$, i.e., every clause in $N^\star$ contains a negative literal that is set to 1, and let $P^\star$ be the set of clauses satisfied only positively by $\Psi^\star$, i.e., every clause in $P^\star$ contains a positive literal that is set to 1 and no negative literal in this clause is set to 1. Then, there exists a set $V_{P^\star} \subseteq \mathsf{var}(P^\star)$ of size at most k that satisfies all the clauses in $P^\star$ positively, i.e., for each clause in $P^\star$, there is a variable in $V_{P^\star}$ that occurs as a positive literal in it and is assigned 1 under $\Psi^\star$. Let Ψ be a good assignment which is generated with probability at least $(\frac{\varepsilon}{2r})^k$. Since Ψ is a good assignment, $T(\Psi^\star) \subseteq T(\Psi)$ and $F(\Psi) \subseteq F(\Psi^\star)$. Hence, $P^\star \subseteq P$. Thus, $P^\star \subseteq \mathcal{U}$ and for each variable in $\mathsf{var}(P^\star)$, we have a set in the family $\mathcal{F}$. Let $Z = \{f_v \in \mathcal{F} \colon v \in V_{P^\star}\}$. We claim that Z is a solution to $\mathcal{J}_\Psi$. Clearly, Z covers at least $|P_i^\star|$ elements in $\mathcal{U}$, for each $i \in [r]$. We claim that $t_i' \leq |P_i^\star|$. Since $\Psi^\star$ is a solution to $\mathcal{I}$, it satisfies t_i clauses for each $i \in [r]$. Since $t_i = |P_i^\star| + |N_i^\star|$, due to Claim 3, we know that $t_i \leq |P_i^\star| + \frac{|N_i|}{1-\varepsilon}$. Thus, $|P_i^\star| \geq t_i - \frac{|N_i|}{1-\varepsilon}$. Since $t_i' = t_i - \frac{|N_i|}{1-\varepsilon}$, $t_i' \leq |P_i^\star|$. As $V_{P^\star}$ is an

independent set in the matroid $\mathcal{M}$, we can conclude that Z is also an independent set in the matroid $\mathcal{M}'$. This holds due to our construction. This completes the proof. □

Lemma 8. *Assume that $\mathcal{I} = (\Phi, r, \chi, \mathcal{M}, k, t)$ is a yes-instance of* (M, F)-Max k-weight SAT. *If there exists $(1-\varepsilon)$-approximate solution for $\mathcal{J}_\Psi = ((\mathcal{U}, \mathcal{F}), r', \chi', \mathcal{M}', k', t')$, where Ψ is a good assignment, then there exists $(1-\varepsilon)$-approximate solution for $\mathcal{I}$ with probability at least $\frac{1}{2}$.*

Proof. Let $\Psi^\star$ be an optimal assignment. Due to Claim 3, Ψ satisfies at least $(1-\varepsilon)|N_i^\star|$ clauses negatively, for each $i \in [r]$, with probability at least $1/2$. Let S be a $(1-\varepsilon)$-approximate solution to $\mathcal{J}_\Psi$. We construct an assignment σ as follows: if $f_x \in S$, then $\sigma(x) = 1$, otherwise 0. We claim that σ is a $(1-\varepsilon)$-approximate solution to $\mathcal{I}$. As $S \in I'$, it is clear that $\sigma^{-1}(1) \in I$. This holds due to our construction. Moreover, due to the construction of $\mathcal{J}_\Psi$, note that $\mathcal{F}$ does not contain a set corresponding to the variable that is set to 0 by Ψ. Thus, if $\Psi(x) = 0$, then $\sigma(x) = 0$. Hence, σ satisfies at least $(1-\varepsilon)|N_i^\star|$ clauses negatively, for each $i \in [r]$, with probability at least $1/2$. Next, we argue that σ satisfies at least $(1-\varepsilon)|P_i^\star|$ clauses only positively, for each $i \in [r]$. Since S is a $(1-\varepsilon)$-approximate solution to $\mathcal{J}_\Psi$, for each $i \in [r]$, S covers at least $(1-\varepsilon)t_i'$ elements. Recall that $\mathcal{U}$ contains an element corresponding to each clause in $\bigcup_{i\in[r]} P_i$. Thus, σ satisfies at least $(1-\varepsilon)t_i'$ clauses only positively for each $i \in [r]$. Recall that $t_i' = t_i - \frac{|N_i|}{1-\varepsilon}$. Thus, $|P_i| + |N_i| \geq (1-\varepsilon)(t_i - \frac{|N_i|}{1-\varepsilon}) + |N_i| = (1-\varepsilon)t_i$. Hence, σ is a $(1-\varepsilon)$-approximate solution for $\mathcal{I}$. □

Due to Lemmata 7 and 8, we have the following result.

Theorem 4. *There exists a polynomial-time randomized algorithm that given a* yes *instance $\mathcal{I}$ of* (M, F)-Max k-weight SAT *generates a* yes *instance $\mathcal{J}$ of* (M, F)-Maximum Coverage *with probability at least $(\frac{\varepsilon}{2r})^k$. Furthermore, given a factor $(1-\varepsilon)$-approximate solution of $\mathcal{J}$, it can be extended to a $(1-\varepsilon)$-approximate solution of $\mathcal{I}$ with probability at least $1/2$.*

Remark 2. *Note that if the variable-clause incidence graph of the input formula belongs to a subgraph closed family $\mathcal{H}$, then the incidence graph of the resulting instance of* (M, F)-Maximum Coverage *will also belong to $\mathcal{H}$.*

Due to Theorem 4 and Theorem 3, we have the following result, which is an improvement over Theorem 1.

Theorem 5. *There is a randomized algorithm that given a* yes *instance* $\mathcal{I}$ *of* (M, F)-Max k-weight SAT*, where the variable-clause incidence graph is* $K_{d,d}$*-free, returns a factor* $(1-\varepsilon)$*-approximate solution with probability at least* $(1-\frac{1}{e})$*, and runs in time* $(\frac{rdk}{\varepsilon})^{\mathcal{O}(dk)}(n+m)^{\mathcal{O}(1)}$*.*

The $(\frac{r}{\varepsilon})^{\mathcal{O}(k)}$ factor in the running time of the algorithm in Theorem 5 comes by repeating the algorithm in Theorem 4, followed by Theorem 3, independently $(\frac{r}{\varepsilon})^{\mathcal{O}(k)}$ many times. This also boosts the success probabilty to at least $(1-\frac{1}{e})$.

Chapter 7

Fairness Constrained Partial Coverage and Satisfiability

In Chapter 6, we established the equivalence of CC-MaxSat and Maximum Coverage with respect to EPAS parameterized by k via a reduction algorithm. The reduction enables us to direct our focus solely on Maximum Coverage while designing an EPAS as, combined with the reduction, it yields an EPAS for CC-MaxSat incurring only a multiplicative $\left(\frac{1}{\varepsilon}\right)^{\mathcal{O}(k)}$ factor. In this chapter, we give EPAS for two generalizations of Maximum Coverage: F-Maximum Coverage on $K_{d,d}$-free set systems and a more efficient one for F-Maximum Coverage on bounded frequency set systems.

Equivalent reformulation in terms of dominating set in the incidence graph. For the ease of exposition, we recast F-Maximum Coverage to the following problem, and work with this formulation.

Partition Color Constrained Dominating Set (PCCDS)
Input: An instance $\mathcal{I} = (G, r, f, k, t)$, where $G = (A \uplus B, E)$ is a bipartite graph, $f : B \to [r]$ is a surjective function, a non-negative integer k, and for each color $j \in [r]$, a *coverage requirement* $t_j \geq 0$.
Here, $B_j \coloneqq \{\mathfrak{s} \in B : f(\mathfrak{s}) = j\}$ is a set of vertices of *color* j, for each $j \in [r]$.
Parameter: k
Question: Does there exist a subset $S \subseteq A$, such that (1) $|S| \leq k$, and (2) for each $j \in [r]$, $|N_j(S)| \geq t(j)$? Here, $N_j(S) \subseteq B_j$ is the set of vertices of color $j \in [r]$ that are adjacent to at least one vertex in S.

Note that PCCDS is the "multiple coverage" version of the well-studied problem Red Blue Dominating Set. We avoid "Red Blue" here just to avoid confusion with our color classes. Observe that finding a subfamily of size k that covers at least t_j elements of the color j is equivalent to finding a set $S \subseteq A$ of size k such that $N_j(S)$ (neighbors of S that are colored j) is at least t_j in the incidence graph.

We first design the Bucketing subroutine in Section 7.2. Then, in Section 7.3, we design and analyze the EPAS for PCCDS when each vertex in B has degree at most d (i.e., frequency at most d). Finally, we give an EPAS for PCCDS on $K_{d,d}$-free graphs in 7.4.

7.1 Overview of EPAS in the presence of constraints

Deterministic and Randomized Branching using a Largest Set

As a warm-up, we start with the vanilla Maximum Coverage on frequency-d set systems (where our algorithms *do not* improve over the known algorithms in the literature), and give a complete formal proof. Then, we will gradually introduce the ideas required to handle fairness and matroid constraints – first separately, and then together. Finally, we briefly discuss the ideas required to these results to $K_{d,d}$-free set systems and multiple matroid constraints.

To introduce our ideas in a clean and gradual way, we start with the simplest setting of Maximum Coverage where the maximum frequency of the elements is bounded by d. Recall that we are given an instance $(\mathcal{U}, \mathcal{F}, k)$ and the goal is to find a sub-family of $\mathcal{F}$ of size k that covers the maximum number of elements. For any sub-family $\mathcal{R} \subseteq \mathcal{F}$, let $\mathcal{U}(\mathcal{R})$ denote the subset of elements covered by $\mathcal{R}$, and $\mathsf{OPT}_k(\mathcal{R})$ denote the maximum number of elements that can be covered by a subset of $\mathcal{R}$ of size k. Further, for a set $S \in \mathcal{F}$, we denote by $\mathcal{F} - S$, the family obtained by removing S, as well as the elements of S from each of the remaining sets. Our approach is inspired by the approaches of Skowron and Faliszewski [24] and Manurangsi [19] who show that $\mathcal{O}(kd/\varepsilon)$ sets of the largest size is guaranteed to contain a $(1-\varepsilon)$-approximate solution. This naturally begs the question, "*why not start by adding the largest set into the solution?*". Let us inspect this question more closely. Let L be a largest set in $\mathcal{F}$. By looking at the contribution of coverage of each set in an optimal solution, say $\mathcal{O}$, we can easily see that $|L| \geq \frac{\mathsf{OPT}_k(\mathcal{F})}{k}$. We

say that a set $S \in \mathcal{F}$ is *heavy* w.r.t. L if $|L \cap S| \geq \frac{\varepsilon|L|}{k}$ (note that L is heavy w.r.t. itself). However, since the frequency of each element is bounded by d, each element in L can appear in at most d (in fact, $d-1$) sets $L \cap S$ for different $R \in \mathcal{F}$. This implies that at most $\frac{kd}{\varepsilon}$ sets in $\mathcal{F}$ are heavy w.r.t. L.

Algorithm. Our algorithm simply branches on the sets in $\mathcal{H}(L)$, which is the family of heavy sets w.r.t. L. Specifically, in the branch corresponding to a heavy set $S \in \mathcal{H}(L)$, we include it in the solution, and recursively call the algorithm on the residual instance $(U \setminus S, \mathcal{F} - S, k-1)$. If any of the sets in $\mathcal{O}$ is heavy w.r.t. L, then in the branch corresponding to such a set yields an approximate solution via induction. The main idea is that, if no set in $\mathcal{O}$ is heavy w.r.t. L, then the branch corresponding to L yields a good solution. This is justified as the effect of any of the $k-1$ sets in $\mathcal{O}$ is too small. For the sake of clarity, we formally analyze this algorithm below via induction.

We want to show that, for a given input $(\mathcal{U}', \mathcal{F}', k')$ the recursive algorithm returns a family $\mathcal{R} \subseteq \mathcal{F}'$ of size k' such that $|\mathcal{U}'(\mathcal{R})| \geq (1-\varepsilon) \cdot \mathsf{OPT}_{k'}(\mathcal{F}')$. The base case for $k' = 0$ is trivial, since the algorithm returns an empty set. Suppose that the claim is true for all inputs with budget $k-1$, and we want to prove it for $(\mathcal{U}, \mathcal{F}, k)$. Let $\mathcal{O}$ denote an optimal solution of size k with $|\mathsf{OPT}_k(\mathcal{F})| = |U(\mathcal{O})|$.

Approximation Ratio in the Easy Case. If $\mathcal{O} \cap \mathcal{H}(L) \neq \emptyset$, then there exists a branch corresponding to a set $S \in \mathcal{O} \cap \mathcal{H}(L)$. This is the *easy case* (for the analysis). In this case, $\mathsf{OPT}_{k-1}(\mathcal{F} - S) = \mathsf{OPT}_k(\mathcal{F}) - |S|$, and hence the approximation ratio is:

$$\begin{aligned} \frac{|S| + (1-\varepsilon)\mathsf{OPT}_{k-1}(\mathcal{F} \setminus S)}{\mathsf{OPT}_k(\mathcal{F})} &= \frac{|S| + (1-\varepsilon)(\mathsf{OPT}_k(\mathcal{F}) - |S|)}{\mathsf{OPT}_k(\mathcal{F})} \\ &\geq \frac{(1-\varepsilon)(\mathsf{OPT}_k(\mathcal{F})}{\mathsf{OPT}_k(\mathcal{F})} = (1-\varepsilon) \end{aligned}$$

Approximation Ratio in the Hard Case. The more hard case (for analysis) is when $\mathcal{O} \cap \mathcal{H}(S) = \emptyset$. In this case, we argue that the branch that includes the element L is good enough. As in the easy case, we first lower bound the value of $\mathsf{OPT}_{k-1}(\mathcal{F} - L)$. By counting the unique contributions to the solution, there exists a *light* set $S_l \in \mathcal{O}$ such that for $\mathcal{O}' = \mathcal{O} \setminus \{S_l\}$, it holds $|U'(\mathcal{O}')| \geq \frac{k-1}{k} \cdot \mathsf{OPT}_k(\mathcal{F})$. Because no set in $\mathcal{O}$ is heavy w.r.t. L, it follows that for each $R \in \mathcal{O}'$, it holds that $|R \cap L| < \frac{\varepsilon|L|}{k}$, and therefore by counting it holds that $|U(\mathcal{O}') \cap L| < \varepsilon \cdot |L|$.

Therefore,

$$\mathsf{OPT}_{k-1}(\mathcal{F} \setminus L) \geq |U'(\mathcal{O}') \setminus L| \geq |U(\mathcal{O}')| - |U(\mathcal{O}') \cap L| \geq \frac{k-1}{k} \cdot \mathsf{OPT}_k(\mathcal{F}) - \varepsilon \cdot |L|.$$

Therefore, the approximation ratio of the branch that includes L is as follows.

$$\begin{aligned}
\frac{|L| + (1-\varepsilon)\mathsf{OPT}_{k-1}(\mathcal{F} - L)}{\mathsf{OPT}_k(\mathcal{F})} &= \frac{|L| + (1-\varepsilon)\left(\frac{k-1}{k} \cdot \mathsf{OPT}_k(\mathcal{F}) - \varepsilon \cdot |L|\right)}{\mathsf{OPT}_k(\mathcal{F})} \\
&\geq \frac{|L| + (1-\varepsilon)\left(\frac{k-1}{k} \cdot \mathsf{OPT}_k(\mathcal{F})\right) - \varepsilon|L|}{\mathsf{OPT}_k(\mathcal{F})} \\
&= \frac{(1-\varepsilon)|L| + (1-\varepsilon)\left(\frac{k-1}{k} \cdot \mathsf{OPT}_k(\mathcal{F})\right)}{\mathsf{OPT}_k(\mathcal{F})} \\
&\geq \frac{(1-\varepsilon)(\mathsf{OPT}_k(\mathcal{F})}{\mathsf{OPT}_k(\mathcal{F})} = (1-\varepsilon)
\end{aligned}$$

The second last inequality holds from the fact that $|L| \geq \frac{\mathsf{OPT}_k(\mathcal{F})}{k}$.

This leads to a deterministic $(1-\varepsilon)$-approximation algorithm with running time $((\frac{kd}{\varepsilon})^k) \cdot (n+m)^{\mathcal{O}(1)}$.

Insight into the probabilistic branching. A closer inspection of the analysis reveals that the reason L may not a good choice is that the sets of $\mathcal{O}$ *together* cover more than a certain threshold fraction of elements covered by L. We utilize this idea through a smoothening process that captures the effect of the size of the intersection of a set S with L in a more nuanced manner. Let us define the weight $h_L(S)$ of a set $S \in \mathcal{F} \setminus \{L\}$ as $h_L(S) = \frac{|S \cap L|}{|L|}$. Our algorithm now instead does "randomized branching", i.e., it samples one set to be included in the solution according to some probability distribution, and then continues recursively. Note that the single run of the algorithm finishes in polynomial time. The probability distribution used by the algorithm is as follows: the set L is sampled with probability $1/2$, and any other set $S \in \mathcal{F} \setminus \{L\}$ is sampled with probability proportional to its weight $h(S)$ (the constant of proportionality is chosen such that this is a valid probability distribution, that is, the probabilities sum up to 1). In particular, observe that $\sum_{S \in \mathcal{F} \setminus \{L\}} h_L(S) \leq \frac{d|L|}{|L|} = d$. Thus, the probability of selecting S is at least $\frac{h_L(S)}{2d}$. Note that due to the way the weights $h_L(S)$ are defined, the sets with a large intersection with L have a greater chance of being sampled, as compared to the sets with a small intersection with L.

We will show that the algorithm returns a $(1-\varepsilon)$-approximate solution with probability at least $(\frac{\varepsilon}{2d})^k$, and runs in polynomial time. This implies that by repeating the algorithm $(\frac{2d}{\varepsilon})^k$ times, we obtain a $(1-\varepsilon)$-approximation with probability at least a positive constant. This leads to a randomized algorithm with running time $\mathcal{O}^*((\frac{2d}{\varepsilon})^k)$.

The proof is again by induction. We want to show that, for any input $(U', \mathcal{F}', k')$, the algorithm returns a solution $\mathcal{R} \subseteq \mathcal{F}'$ of size k' such that $|U'(\mathcal{R})| \geq (1-\varepsilon) \cdot \mathsf{OPT}_{k'}(\mathcal{F}')$, with probability at least $(\frac{\varepsilon}{2d})^{k'}$. We reuse much of the notation from the previous analysis. Let $\mathcal{O}$ be an optimal solution of size k. First, the case when $L \in \mathcal{O}$, since our algorithm samples and includes L in the solution with probability $1/2$. Then, conditioned on this event (that is, L being sampled), the approximation ratio analysis proceeds similarly to the *easy case* of the previous analysis. By induction, the recursive algorithm returns a $(1-\varepsilon)$-approximate solution with probability at least $(\frac{\varepsilon}{2d})^{k-1}$. Thus, overall, the algorithm returns a $(1-\varepsilon)$-approximation with probability at least $\frac{1}{2}(\frac{\varepsilon}{2d})^{k-1} \geq (\frac{\varepsilon}{2d})^k$.

Now suppose $L \notin \mathcal{O}$. As before, let $S_l \in \mathcal{O}$ be a light set as defined earlier, and $\mathcal{O}' = \mathcal{O} \setminus \{S_l\}$. We analyze by considering the following two cases: either (i) $|U(\mathcal{O}') \cap L| \leq \varepsilon \cdot |L|$, or (ii) $|U(\mathcal{O}') \cap L| > \varepsilon \cdot |L|$.

In case (i), we are effectively in the same situation as the *hard case* of the previous analysis – as before, the algorithm samples L with probability at least $1/2$, and as argued in the *hard case*, conditioned on the previous event, the branch corresponding to L returns a $(1-\varepsilon)$-approximate solution, but now with probability at least $(\frac{\varepsilon}{2d})^{k-1}$ by induction. Therefore, we obtain an $(1-\varepsilon)$-approximate solution with probability at least $\frac{1}{2}(\frac{\varepsilon}{2d})^{k-1} \geq (\frac{\varepsilon}{2d})^k$.

In case (ii), we have that $|U(\mathcal{O}') \cap L| > \varepsilon \cdot |L|$. Notice that,

$$\sum_{S \in \mathcal{O}'} |S \cap L| > \varepsilon.$$

This implies that

$$\sum_{S \in \mathcal{O}'} h_L(S) = \sum_{S \in \mathcal{O}'} \frac{|S \cap L|}{|L|} \geq |U(\mathcal{O}') \cap L| > \varepsilon |L|.$$

Therefore, the total weight of the sets in $\mathcal{O}'$ is at least ε. Therefore, the probability that the algorithm samples a set from $\mathcal{O}'$ is at least $\frac{\varepsilon}{2d}$. Conditioned on this event, the approximation ratio analysis now proceeds as in the *easy case*, and the algorithm

returns a $(1-\varepsilon)$-approximate solution with probability at least $\frac{\varepsilon}{2d}(\frac{\varepsilon}{2d})^{k-1} = (\frac{\varepsilon}{2d})^k$.

Handling fairness constraints via the Bucketing trick

The aforementioned idea of prioritizing the largest set L, or sets that are heavy w.r.t. it, fails to generalize when we have multiple coverage constraints in F-Maximum Coverage. This is simply because there is no notion of "the largest set" even when we want to cover elements of *two* different colors, each with different coverage requirements. To handle such multiple coverage constraints, our idea is to use multidimensional-knapsack-style bucketing technique to group the sets of $\mathcal{F}$ into *approximate equivalence classes*, called *bags* for short. At a high level, all the vertices belonging to a particular bag contain *approximately equal* (i.e., within a factor of $(1+\varepsilon)$) number of elements of *all* of the r colors. Thus, in isolation, any two sets L_1 and L_2 belonging to the same bag are interchangeable, since we tolerate an ε-factor loss in the coverage. Since the total number of bags can be shown to be $\left(\frac{\log k}{\varepsilon}\right)^{\mathcal{O}(rk)}$, and hence we can "guess" a bag $\mathcal{B}$ containing a set from a solution $\mathcal{O}$. However, due to different amount of overlap with an optimal solution $\mathcal{O}$, two sets $L_1, L_2 \in \mathcal{B}$ may not be interchangeable w.r.t. $\mathcal{O}$. That is, if $L_1 \in \mathcal{O}$, then $\mathcal{O} \setminus \{L_1\} \cup \{L_2\}$ may not be a good solution. However, assuming that we have correctly guessed a bag that intersects with $\mathcal{O}$, we can then select a set $L \in \mathcal{B}$, and define the heavy sets (for deterministic algorithm) or weights $h_L(\cdot)$ (for the randomized algorithm) w.r.t. L. Note that, since we have multiple coverage constraints, we cannot simply look at the total size of the intersection $|S \cap L|$. Instead, we need to tweak the notion of *heaviness* that takes into account the number of elements of each color in the intersection $S \cap L$. To summarize, we need two additional ideas to handle multiple colors in F-Maximum Coverage: (1) "guessing" over buckets, and (2) a suitable generalization of the notion of heaviness. Modulo this, the rest of the analysis is again similar to the *easy* and *hard* cases as before.

Further Extensions

The ideas mentioned in the previous subsections can be extended to even more general settings in a couple of ways. First, we describe how to extend the ideas from frequency-d set systems for Maximum Coverage to $K_{d,d}$-free set systems, i.e., set system $(U, \mathcal{F})$, where no d sets in $\mathcal{F}$ contain d elements of U in common. Note that frequency-d set systems are $K_{d+1,d+1}$-free. Then, we describe how the linear algebraic toolkit of *representative sets* can be used to handle multiple (linear)

matroid constraints.

$K_{d,d}$-free Set Systems. Next, we consider $K_{d,d}$-free set systems $(U, \mathcal{F})$, where no d sets of $\mathcal{F}$ contain d common elements of U. To design the EPAS on $K_{d,d}$-free set systems, we combine the bucketing idea along with the combinatorial properties of $K_{d,d}$-free graphs to bound the number of heavy neighbors of a set. To this end, however, we need to modify the precise definition of *heaviness*. This leads to a somewhat cumbersome branching algorithm that handles colors differently based on their coverage requirement. For colors with small coverage requirement, we highlight the covered vertices using the standard technique of *label coding*[1]. Now, vertices in a bag cover the elements with the same label and for colors with high coverage requirements, the sizes of the sets in the same bag are "almost" the same. Then, we pick an *arbitrary* set L from the bag $\mathcal{B}$, and we branch on (suitably defined) *heavy* sets w.r.t. S. Since the number of heavy sets is bounded by a function of k, d, and ε, this leads to a deterministic version of EPAS for F-Maximum Coverage to $K_{d,d}$-free set systems. Note that since frequency-d set systems is a special case of this, this implies a deterministic EPAS in this case; however with a much worse running time compared to F-Maximum Coverage to $K_{d,d}$-free set systems.

7.2 The Bucketing Subroutine.

We design a subroutine, called Bucketing, which (or slight variations thereof) will be crucially used to design EPAS in the subsequent sections and chapters. We divide the set of vertices of A into a bounded number of equivalence classes depending on the size of their j-neighborhoods. Fix a color j. At a high level, two vertices u and v belonging to a particular equivalence class will have j-degrees that are *approximately equal*. There are two exceptions to this: (A) For a color j, all vertices of degree at least t_j are treated as equivalent as far as color j is concerned (i.e., we still classify such vertices based on their j'-degrees for other colors j') – since any single vertex from such a class is sufficient to entirely take care of color j. (B) all vertices of j-degree less than $\frac{\varepsilon t_j}{2k}$ are treated as equivalent as far as color j is concerned, for the following reason. The difference between the j-degrees between such two vertices is at most $\frac{\varepsilon t_j}{2k}$. Thus, even if we make a bad choice at most k times, we only lose

[1]The technique is better known as *color coding*. However, this creates an unfortunate clash of terminology – these *colors* have nothing to do with the original colors corresponding to coverage constraints. Bandyapadhyay et al. [35] instead use the term "label coding", and we also adopt the same terminology

at most εt_j coverage for the color j. Thus, the *interesting range* of j-degrees is between $\left[\frac{\varepsilon t_j}{2k}, t_j\right]$, which is sub-divided into intervals of range $(1+\varepsilon)$. It is easy to see that the vertices are partitioned into $\mathcal{O}(\log_{1+\varepsilon} k)$ classes for a particular color. We proceed in a similar manner for each color $j \in [r]$, and then we obtain our final set of equivalence classes, such that all the vertices in a particular class are *equivalent* w.r.t. each color in the sense described above.

Our algorithm for PCCDS is recursive, and will use the Bucketing procedure as a subroutine. During the course of the recursive algorithm, we may modify the instance in a variety of ways – delete a subset of vertices, restrict the coverage function t, decrement the value of k, or delete a subset of colors. However, in the Bucketing subroutine we require the *original* value of k, which we will denote by k^*. Now, we formally state the procedure.

Bucketing procedure.
Let $\mathcal{I} = (G = (A \uplus B, E), [r], f, t, k)$ be the *current* instance of PCCDS. Then, let $\lambda \coloneqq \lceil \log_{(1+\varepsilon)} \frac{2k^*}{\varepsilon} \rceil$. Fix a color $j \in [r]$. For every $1 \leq \alpha \leq \lambda$, we define

$$A(j,\alpha) \coloneqq \left\{ v \in A : \frac{t_j}{(1+\varepsilon)^{\alpha}} \leq d^j(v) < \frac{t_j}{(1+\varepsilon)^{\alpha-1}} \right\}.$$

We also define $A(j,0) \coloneqq \{v \in A : d_j(v) \geq t_j\}$, and $A(j,\lambda{+}1) \coloneqq \left\{ v \in A : d_j(v) < \frac{t_j}{(1+\varepsilon)^{\lambda}} \right\}$.

Let $\mathbf{V} = \{0, 1, \ldots, \lambda, \lambda+1\}^r$, and consider an arbitrary vector $\mathbf{v} \in \mathbf{V}$. Let $\mathbf{v} = (\alpha_1, \alpha_2, \ldots, \alpha_r)$. Then, we define $A(\mathbf{v}) \coloneqq \bigcap_{j=1}^{r} A(j, \alpha_j)$. We call any such $A(\mathbf{v})$ as a *bag*. The Bucketing procedure first computes the set of bags as defined above, and returns only the set of non-empty bags, which form a partition of A. It is easy to see that the procedure can be implemented in polynomial time.

Observation 1. *The items 1-3 in the following hold for any color $j \in [r]$.*

1. *For any $v \in A(j, \lambda+1)$, $d_j(v) < \frac{\varepsilon t_j}{2k^*}$.*
2. *For any $v \in A(j,0)$, $d_j(v) \geq t_j$.*
3. *For any $1 \leq \alpha \leq \lambda$, and for any $u, v \in A(j,\alpha)$, it holds that $\frac{d_j(u)}{(1+\varepsilon)} \leq d_j(v) \leq (1+\varepsilon) \cdot d_j(u)$.*

The number of bags returned by Bucketing *is bounded by*

$$\prod_{j=1}^{r} (\lambda + 2) \leq \left(\frac{6 \log k^*}{\varepsilon^2} \right)^r =: L.$$

7.3 An EPAS for PCCDS with Bounded Frequency

Our algorithm for PCCDS, when each vertex in B has degree at most d is given in Algorithm 3. This algorithm is recursive, and takes as input an instance $\mathcal{I}$ of PCCDS; initially the algorithm is called on the original instance. We assume that we remember the value of k^* from the original instance, since that is used in the BUCKETING subroutine. Now we discuss the algorithm. In the algorithm, first we use the BUCKETING subroutine to partition the vertices of A into a number of equivalence classes. Then, we pick one of the bags uniformly at random – essentially, in this step we are trying to guess the bag that has a non-empty intersection with a hypothetical solution S. [2]

Suppose we correctly guess such a bag $A(\mathbf{v})$. Then, we arbitrarily pick a vertex v from this bag, and use it to define a probability distribution on all the vertices in A. This distribution places a constant ($\geq 1/2$) probability mass on sampling v, and the rest of the probability mass is split proportional to the number of common neighbors of a vertex w with v. Then, we sample a vertex u according to this distribution, add it to the solution, and recurse on the residual instance. Here, the intuition is that, if the vertices in S have a lot of common neighbors with v, then one of the vertices from S will be sampled with reasonably large probability. In this case, we recurse on a vertex from a hypothetical solution, i.e., a "correct choice". Otherwise, if the vertices in S have very few common neighbors with v, then we claim that we can replace a vertex in $S \cap A(\mathbf{v})$ with v, and still obtain a good (hypothetical) solution to compare against. In this case, we argue that v is the "correct choice". Now, we state the algorithm formally, and then proceed to the analysis.

First we state the following observation.

Observation 2. $\sum_{w \in A} p(w) \leq 1$.

Proof. If v is the vertex chosen in Step 5 of the Algorithm 3, then $p(v) := 1/2$.

[2] We note that this step can be replaced by deterministically branching on each of the bags instead; but since the next step is inherently randomized, we continue with the current presentation for the simplicity of exposition.

Algorithm 3 PCCDS($\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t)$)

if $k = 0$ **then**
  return $\emptyset$
end if
Let $\mathcal{A}$ be the set of bags returned by applying BUCKETING on $\mathcal{I}$
Choose a bag $A(\mathbf{v}) \in \mathcal{A}$ uniformly at random, and select an arbitrary vertex $v \in A(\mathbf{v})$
Define the following quantities w.r.t. the vertex v:
  • For any $w \in A \setminus v$ and any color j, let $h_j(w) := \frac{|N_j^G(w) \cap N_j^G(v)|}{|N_j^G(v)|}$,
  • Let $p(w) := \frac{1}{2rd} \cdot \sum_{j=1}^{r} h_j(w)$
  • Let $p(v) := \frac{1}{2}$
Sample a vertex $u \in V(G)$ at random proportional to the quantities $p(\cdot)$
$\mathcal{I}' \leftarrow$ PRUNEINSTANCE$(\mathcal{I}, u)$ ▷ Residual instance after adding u to the solution
return $\tilde{S} \cup \{u\}$, where $\tilde{S} \leftarrow$ PCCDS$(\mathcal{I}')$

Therefore, we show that $\sum_{w \in A_1} p(w) \leq \frac{1}{2}$, where $A_1 = A \setminus \{v\}$.

$$\begin{aligned}
\sum_{w \in A_1} p(w) &= \frac{1}{2rd} \sum_{w \in A_1} \sum_{j=1}^{r} h_j(w) \\
&= \frac{1}{2rd} \sum_{j=1}^{r} \frac{1}{|N_j^G(v)|} \sum_{w \in A_1} |N_j^G(w) \cap N_j^G(v)| \\
&\leq \frac{1}{2rd} \sum_{j=1}^{r} \frac{1}{|N_j^G(v)|} \cdot d \cdot |N_j^G(v)| = \frac{1}{2}
\end{aligned}$$

Where, the second-last inequality follows from the fact that every vertex in $N_j^G(v)$ is counted at most d times in $\sum_{w \in A_1} |N_j^G(w) \cap N_j(v)|$, since every vertex in B has degree at most d. □

Now we explain how to sample a vertex proportional to the quantities $p(\cdot)$. Note that Observation 2 implies that $\ell := \sum_{w \in A} p(w) \leq 1$. Also note that $\ell \geq p(v) = 1/2$. Now, we sample a vertex from A such that the probability of sampling a vertex w is equal to $p(w)/\ell$ – this can be done, e.g., by mapping the vertices to disjoint sub-intervals of $[0, 1]$ of length $p(w)/\ell$, and then sampling from uniform distribution over $[0, 1]$. Note that the sum of probabilities is equal to 1, and thus this is a valid probability distribution. Finally, observe that for any set $W \subseteq A$ of vertices, the probability that a vertex from W is sampled is equal to $\sum_{w \in W} p(w)/\ell \geq \sum_{w \in W} p(w)$. Next we prove the correctness of our algorithm.

Lemma 9. *Consider a recursive call PCCDS($\mathcal{I}$), where $\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t)$,*

procedure PRUNEINSTANCE($\mathcal{I}, u$) $\quad \mathcal{I} = (G = (A \uplus B, E), [r], f, k, t)$ and $u \in A$

1: Let $J \coloneqq \{j \in [r] : |N_j(u)| \geq t_j\}$ ▷ J is the set of colors already satisfied by u
2: **if** $J = \emptyset$ **then**
3: $\quad G' \coloneqq G \setminus\setminus u$ ▷ Remove u from A and all of its neighbors from B
4: $\quad f'$ is the restriction of f to $B \setminus \bigcup_{j \in [r]} N_j(u)$
5: $\quad t'_j = t_j - |N_j(u)|$ for all colors j ▷ Account for addition of u into the solution
6: $\quad \mathcal{I}' = (G', [r], f', k-1, t')$
7: **else**
8: $\quad r' \coloneqq r - |J|$, and w.l.o.g. assume that $J = \{r'+1, r'+2, \ldots, r\}$
9: $\quad G' = G[A' \uplus B']$, where $A' \coloneqq A \setminus \{u\}$, and $B' \coloneqq B \setminus \left(N(u) \cup \bigcup_{j \in J} f^{-1}(j)\right)$
▷ Remove u, its neighbors in B, and vertices of all the colors satisfied by u
10: $\quad f'$ is the restriction of f to B', and $t'_j = t_j - |N_j(u)|$ for $j \in [r']$
11: $\quad \mathcal{I}' = (G', [r'], f', k-1, t')$
12: **end if**
13: **return** $\mathcal{I}'$

and let k^* be the value of k from the original instance. *Consider a set $S \subseteq A$ of size k such that, for each $j \in [r]$, $|N_j^G(S)| = \widetilde{t}_j \geq t_j$.*

Then, with probability at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^k$, the algorithm returns a subset $S' \subseteq A$ of size at most k, such that for each $j \in [r]$, $|N_j^G(S')| = t''_j$, where

$$t''_j \geq (1 - 2\varepsilon) \min(\widetilde{t}_j, t_j) - \frac{\varepsilon k}{k^*} \cdot t_j.$$

Proof. We prove this by induction. When $k = 0$, then there is nothing to prove. Now suppose the claim is true for $k - 1 \geq 0$ and we prove it for k.

Fix a set $S \subseteq V(G)$ of size k such that $|N_j^G(S)| = \widetilde{t}_j$ for each $j \in [r]$. There exists a bag $A(\mathbf{v})$ such that $S \cap A(\mathbf{v}) \neq \emptyset$. Since in the first step (step 5), the algorithm picks a bag uniformly at random from at most L bags, the probability that $A(\mathbf{v})$ is picked is at least $\frac{1}{L}$. We condition on the event that this choice is made, and proceed as follows. Suppose the algorithm picks $v \in A(\mathbf{v})$ in step 5 of Algorithm 3. We start with a relatively straightforward observation.

Observation 3. *Consider a vertex $u \in S$, and consider calling* PRUNEINSTANCE$(\mathcal{I}, u)$. *Let J be the set as defined in Step 1 in this procedure, and $\mathcal{I}'$ be the instance that the call would return. Then,*

1. *For any $j \in J$, $|N_j^G(u)| \geq t_j$,*

2. *For any $j \notin J$, in the instance $\mathcal{I}'$, it holds that $t'_j = t_j - |N_j^G(u)| \leq t_j$, and*

3. *For any $j \notin J$, in the instance $\mathcal{I}'$, it holds that $|N_j^{G'}(S \setminus \{u\})| = \widetilde{t}_j - |N_j^G(u)|$.*

Proof. The first item follows from the definition of $N_j^G(u)$. The second item follows from the definition of t'. For the third item, we note that for any color $j \notin J$, the vertices in $N_j^G(u)$ are removed from color j in the instance $\mathcal{I}'$. Thus, $N_j^{G'}(S \setminus \{u\})$ and $N_j^G(u)$ are disjoint sets in $V(G)$, and their sizes add up to $\widetilde{t}_j$. □

Next, we prove the following technical claim.

Claim 4. *Suppose the random vertex u, selected in Step 10 of Algorithm 3, belongs to the set S. Then, let G' be the graph in the instance $\mathcal{I}'$ obtained in Step 11 of Algorithm 3. Then, with probability at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k-1}$, it holds that for any $j \in [r]$,*

$$\left|N_j^G\left(\tilde{S} \cup \{u\}\right)\right| \geq (1-2\varepsilon) \cdot \min(\widetilde{t}_j, t_j) - \frac{\varepsilon k}{k^*} t_j \tag{7.1}$$

Proof. Suppose the set J as defined w.r.t. u in Step 1 of PRUNEINSTANCE is non-empty. Then, for any $j \in J$, Observation 3 implies that $|N_j^G(u)| \geq t_j$, which is at least the bound in the lemma. Thus, it suffices to focus on colors in $[r] \setminus J$.

By Observation 3 (item 3), the set $S \setminus \{u\}$ of size $k-1$ is such that for each $j \notin J$, $|N_j^{G'}(S \setminus \{u\})| = \widetilde{t}_j - |N_j^G(u)|$. Thus, by induction hypothesis (i.e., using Lemma 9), with probability at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k-1}$, PCCDS($\mathcal{I}'$) returns a set $\tilde{S}$ of size at most $k-1$ that satisfies the following property: for any $j \notin J$,

$$\left|N_j^{G'}(\tilde{S})\right| \geq (1-2\varepsilon) \cdot \min(\widetilde{t}_j - |N_j^G(u)|,\ t'_j) - \frac{\varepsilon(k-1)}{k^*} t'_j \tag{7.2}$$

Thus, it follows that the set $\tilde{S} \cup \{u\}$ satisfies that, for any $j \notin J$,

$$\begin{aligned}
\left|N_j^G(\tilde{S} \cup \{u\})\right| &= \left|N^{G'}(\tilde{S})\right| + \left|N_j^G(u)\right| \\
&\geq (1-2\varepsilon) \cdot \left(t_j - |N_j^G(u)|\right) - \frac{\varepsilon(k-1)}{k^*} t'_j + |N_j^G(u)| \\
&\geq (1-2\varepsilon) \cdot t_j - \frac{\varepsilon(k-1)}{k^*} t'_j \\
&\geq (1-2\varepsilon) \cdot \min(\widetilde{t}_j, t_j) - \frac{\varepsilon k}{k^*} t_j && \text{(since } t'_j \leq t_j\text{)}
\end{aligned}$$

This concludes the proof of the claim. □

Now we proceed with the inductive step, where we consider different cases.

Case 1. $v \in S$.
In this case, the probability that the randomly chosen vertex u in Step 10 is equal to v, is at least $1/2$. We condition on this event, and using Claim 4, it follows that the set $\tilde{S} \cup \{v\}$ satisfies (7.1) for each $j \in [r]$, with (conditional) probability at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k-1}$. Thus, the unconditional probability that the set $\tilde{S} \cup \{u\}$ satisfies the required property is at least

$$\frac{1}{2} \cdot \left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k-1} \geq \left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k}.$$

Case 2. Suppose $v \notin S$. Now we consider two sub-cases.

Case 2.1: There exists a color j such that $\sum_{w' \in S} |N_j^G(v) \cap N_j^G(w')| \geq \varepsilon \cdot |N_j^G(v)|$. We first claim that the probability that some vertex w from the set S is chosen to be u in line 10 is at least $\frac{\varepsilon}{r}$. Then, conditioned on this event, we will use the induction hypothesis to show the required bound.

Fix a color j satisfying the case assumption (if there are multiple such colors, pick one such color arbitrarily). Then, since color j satisfies the case assumption, it follows that,

$$\sum_{w' \in S} h_j(v) = \sum_{w' \in S} \frac{|N_j^G(w') \cap N_j^G(v)|}{|N_j^G(v)|} \geq \frac{1}{|N_j^G(v)|} \cdot \varepsilon \cdot |N_j^G(v)| = \varepsilon$$

Therefore,

$$\sum_{w' \in S} p(w') = \sum_{w' \in S} \frac{h(w')}{2rd} \geq \frac{1}{2rd} \cdot \sum_{w \in S} h_j(w') \geq \frac{1}{2rd} \cdot \varepsilon$$

Thus, the probability that some $w \in S$ will be chosen as the vertex u is at least $\frac{\varepsilon}{2rd}$. Now, we condition on this event. Then, by using Claim 4 and an argument similar to case 1, the set $\tilde{S} \cup \{w\}$ satisfies (7.1) for each $j \in [r]$, with probability at least

$$\frac{1}{L} \cdot \frac{\varepsilon}{2rd} \cdot \left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k-1} = \left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k}.$$

Case 2.2: For all colors j, $\sum_{w' \in S} |N_j(v) \cap N_j(w')| \leq \varepsilon \cdot |N_j(v)|$. Hence,

$$\left| \left(\bigcup_{w' \in S} N_j(w') \right) \cap N_j(v) \right| \leq \sum_{w' \in S} |N_j(v) \cap N_j(w')| \leq \varepsilon \cdot |N_j(v)|. \tag{7.3}$$

Recall that the probability that the vertex u is equal to v is at least $1/2$, and we condition on this choice. Furthermore, recall that we have conditioned on the event that $A(\mathbf{v}) \cap S \neq \emptyset$, but due to case assumption $v \notin S$. Therefore, there must exist a vertex $w \in A(\mathbf{v}) \cap S$.

In this case, we aim to show that v "approximately plays the role" of $w \in A(\mathbf{v}) \cap S$ in our solution. To this end, we consider different cases for the value of α_j in the vector $\mathbf{v}$. In each of the cases, we condition on the probability that the recursive call returns a set $\tilde{S}$ with the desired properties. Using induction, this happens with probability at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k-1}$. Thus, the unconditional probability is at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^{k}$ as in Case 1. Now we proceed to the analysis of each of the cases, conditioned on the good events.

Case A: $\alpha_j = 0$**.** Since $v \in A(j, 0)$, $|N_j^G(v)| \geq t_j$. Thus, $|N_j^G(\tilde{S} \cup \{v\}| \geq t_j$, which is at least the claimed bound.

Case B: $\alpha_j = \lambda + 1$**.** Since $v, w \in A(j, \lambda + 1)$, $|N_j^G(v)| \leq \frac{\varepsilon t_j}{2k^*}$, and $|N_j^G(w)| \leq \frac{\varepsilon t_j}{2k^*}$. Now we analyze $\left|N^{G'}(S \setminus \{w\})\right|$. By case assumption, it follows that,

$$\begin{aligned}
\left|N_j^{G'}(S \setminus \{w\})\right| &= \widetilde{t}_j - |N_j^G(w)| - \left|\left(\bigcup_{w' \in S} N_j^G(w')\right) \cap N_j^G(v)\right| \qquad (7.4)\\
&\geq \widetilde{t}_j - |N_j^G(w)| - |N_j^G(v)| \\
&\geq \widetilde{t}_j - \frac{2\varepsilon t_j}{2k^*} = \widetilde{t}_j - \frac{\varepsilon t_j}{k^*}.
\end{aligned}$$

Then, by inductive hypothesis, the solution $\tilde{S}$ satisfies the first inequality in the following.

$$\begin{aligned}
\left|N_j^{G'}(\tilde{S})\right| &\geq (1 - 2\varepsilon) \cdot \left(\widetilde{t}_j - \frac{\varepsilon t_j}{k^*}\right) - \frac{\varepsilon(k-1)t'_j}{k^*} \\
&\geq (1 - 2\varepsilon) \cdot \widetilde{t}_j - \frac{\varepsilon t_j}{k^*} - \frac{\varepsilon(k-1)t'_j}{k^*} \\
&\geq (1 - 2\varepsilon) \cdot \widetilde{t}_j - \frac{\varepsilon k}{k^*} t_j \qquad \text{(Since } t'_j \leq t_j\text{)}
\end{aligned}$$

which is at least the claimed bound.

Case C. $1 \leq \alpha_j \leq \lambda$**.**
Since $v, w \in A(j, \alpha_j)$, Observation 1 implies that

$$|N_j^G(w)| \leq (1 + \varepsilon) \cdot |N_j^G(v)| \qquad (7.5)$$

Analogous to (8.4), we have the following.

$$\begin{aligned}\left|N_j^{G'}(S \setminus \{w\})\right| &= \widetilde{t}_j - |N_j^G(w)| - \left|\left(\bigcup_{w' \in S} N_j^G(w')\right) \cap N_j^G(v)\right| \\ &\geq \widetilde{t}_j - |N_j^G(w)| - \varepsilon \cdot |N_j^G(v)| && \text{(From (8.3))} \\ &\geq \widetilde{t}_j - (1+2\varepsilon) \cdot |N_j^G(v)| && \text{(From (8.5))}\end{aligned}$$

Thus, by inductive hypothesis, it holds that,

$$\begin{aligned}|N_j^G(\tilde{S} \cup \{v\})| &= |N_j^G(v)| + |N^{G'}(\tilde{S})| \\ &\geq |N_j^G(v)| + (1-2\varepsilon) \cdot \left(\widetilde{t}_j - (1+2\varepsilon) \cdot |N_j^G(v)|\right) - \frac{\varepsilon(k-1)}{k^*} \cdot t_j \\ &\geq (1-2\varepsilon) \cdot \widetilde{t}_j + 4\varepsilon^2 \cdot |N_j^G(v)| - \frac{\varepsilon(k-1)}{k^*} \cdot t_j \\ &\geq (1-2\varepsilon) \cdot \widetilde{t}_j - \frac{\varepsilon(k-1)}{k^*} \cdot t_j && \text{(since } |N_j^G(v)| \geq 0)\end{aligned}$$

This completes the induction, and thus the proof of the lemma. □

Theorem 6. *There exists a randomized algorithm that runs in time* $\left(\frac{6dr}{\varepsilon}\right)^k \cdot \left(\frac{18 \log k}{\varepsilon^2}\right)^{kr} n^{\mathcal{O}(1)}$, *and given a* yes *instance* $\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t)$ *of* PCCDS, *where each vertex in* B *has degree at most* d, *with probability at least* $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^k$, *returns a subset* $\tilde{S} \subseteq A$ *of size* k *such that* $|N_j^G(\tilde{S})| \geq (1-\varepsilon)t_j$ *for all colors* j.

Next, we conclude with the proof of Theorem 6, which follows from Lemma 9 in a straightforward manner.

Proof of Theorem 6. Let $S \subseteq A$ be a set of size k such that $|N_j^G(S)| = \widetilde{t}_j \geq t_j$ for all colors j. Let $\tilde{S}$ denote the output of PCCDS($\mathcal{I}$). It is easy to see that the algorithm returns a solution in polynomial time. Next, Lemma 9 implies that with probability at least $q = \left(\frac{1}{L} \cdot \frac{\varepsilon}{2rd}\right)^k$, the set $\tilde{S}$ satisfies that, for each $j \in [r]$,

$$|N_j^G(\tilde{S})| = t_j'' \geq (1-2\varepsilon) \cdot t_j - \frac{\varepsilon k}{k^*} t_j = (1-2\varepsilon)t_j - \varepsilon t_j = (1-3\varepsilon) \cdot t_j$$

Here, we use the fact that since $\mathcal{I}$ is the original instance, we have $k^* = k$. Also note that, for any color $j \in [r], \widetilde{t}_j \geq t_j$.

We make $\mathcal{O}(q^{-1} \log n)$ independent calls to PCCDS($\mathcal{I}$), and if in any of the calls, we find a set $\tilde{S}$ with the claimed properties, then we return it. Otherwise, the algorithm concludes that $\mathcal{I}$ is a no instance. We get the claimed running time by rescaling ε to $\varepsilon/3$. □

Corollary 1. *Let $\varepsilon > 0$. Then,* F-MAX k-WEIGHT SAT *admits a randomized* EPAS *with running time* $\left(\frac{6dr}{\varepsilon}\right)^k \cdot \left(\frac{18 \log k}{\varepsilon^2}\right)^{kr} (n+m)^{\mathcal{O}(1)}$ *on formulas when the size of the clauses is bounded by d or when every variable appears in at most d clauses. This result holds with constant probability.*

7.4 An EPAS for PCCDS on $K_{d,d}$-free graphs

We now design an EPAS for PCCDS on $K_{d,d}$-free graphs. We start by proposing a randomized algorithm, which will be derandomized later using the known tool of (p,q)-*perfect hash family* [66, 37].

The algorithm first partitions the set of colors in $[r]$ into two sets based on their coverage requirements (high or low).

- Set of colors with low coverage requirement $\mathsf{T}_{\mathsf{small}} := \{j \in [r] \colon t_j \leq {}^{2k^2d}/_{\varepsilon}\}$
- Set of colors with high coverage requirement$\mathsf{T}_{\mathsf{large}} := \{j \in [r] \colon t_j > {}^{2k^2d}/_{\varepsilon}\}$

Henceforth, we assume that we are given a yes instance and show that the algorithm outputs an approximate solution with *high probability*; otherwise, the algorithm detects that we are given a no instance. Let $S^\star$ be a hypothetical feasible solution, i.e., $S^\star \subseteq A$, $|S^\star| \leq k$ and $|N_j(S^\star)| \geq t_j$, for each $j \in [r]$. Let $B_{\mathsf{small}} = \bigcup_{j \in \mathsf{T}_{\mathsf{small}}} B_j$ be the set of vertices in B that belong to color classes with low coverage requirement.

We use *color coding* to identify the vertices in B_{small} that are covered by $S^\star$ with *high probability*. Without loss of generality, let $\mathsf{T}_{\mathsf{small}} = \{1, \ldots, z\}$, where $z \leq r$. As the requirement for any color in $\mathsf{T}_{\mathsf{small}}$ is at most $\frac{2k^2d}{\varepsilon}$, it suffices to highlight $\sum_{j \in \mathsf{T}_{\mathsf{small}}} t_j \leq \frac{2k^2d}{\varepsilon} \cdot z$ vertices from B_{small} that are covered by $S^\star$.

Separation of small cover: Label the vertices of B_{small} uniformly and independently at random using $\frac{2k^2dz}{\varepsilon}$ labels, say, $1, \ldots, \frac{2k^2dz}{\varepsilon}$.

Note that $S^\star$ may cover more than t_j vertices of color j, however, we are only concerned with identifying any t_j vertices among the ones that are covered. The following proposition gives a lower bound on the success probability.

Proposition 7. [11, Lemma 5.4] *Let $\mathcal{U}$ be a universe and $X \subseteq \mathcal{U}$. Let $\chi \colon \mathcal{U} \to [|X|]$ be a function that colors each element of $\mathcal{U}$ with one of $|X|$ colors uniformly and*

independently at random. Then, the probability that the elements of X are colored with pairwise distinct colors is at least $e^{-|X|}$.

Next, the BUCKETING procedure is implemented on the vertices in A based on their *colored degree*, which we describe in detail below. We do approximate bucketing with respect to the colors in $\mathsf{T}_{\mathsf{large}}$ (this is similar to the procedure mentioned in Section 7.2), while we do exact bucketing with respect to the colors in $\mathsf{T}_{\mathsf{small}}$.

For a vertex $v \in B_{\mathsf{small}}$, let $\mathsf{lab}(v)$ denote its label. For $X \subseteq B_{\mathsf{small}}$, let $\mathsf{lab}(X) = \cup_{v \in X} \mathsf{lab}(v)$. Let $\mathsf{labels} = \{1, \ldots, {}^{2k^2dz}/_{\varepsilon}\}$.

We first create buckets for all $j \in \mathsf{T}_{\mathsf{small}}$ as follows.

BUCKETING(SMALL). For every $j \in \mathsf{T}_{\mathsf{small}}$ and a set $\gamma \subseteq \mathsf{labels}$, we define

$$\mathbb{A}_s(j, \gamma) \coloneqq \{v \in A \colon \mathsf{lab}(N_j(v)) = \gamma\}$$

Consider an arbitrary vector $\mathbf{v} = (\gamma_1, \ldots, \gamma_z)$, where $\gamma_j \subseteq \mathsf{labels}$ for each $j \in \mathsf{T}_{\mathsf{small}}$. Then, $\mathfrak{B}_s(\mathbf{v}) \coloneqq \bigcap_{j=1}^{z} \mathbb{A}_s(j, \gamma_j)$. If $u, w \in \mathfrak{B}_s(\mathbf{v})$, $\mathsf{lab}(N_j(u)) = \mathsf{lab}(N_j(w)) = \gamma_j$, for each $j \in \mathsf{T}_{\mathsf{small}}$.

Next, we create buckets with respect to all the colors in $j \in \mathsf{T}_{\mathsf{large}}$.

BUCKETING(LARGE). Let $\lambda = \lceil \log_{(1+\varepsilon)} \frac{\varepsilon n}{2kd} \rceil$. For every color $j \in \mathsf{T}_{\mathsf{large}}$ and $1 \leq \alpha \leq \lambda$, we define

$$\mathbb{A}_\ell(j, \alpha) \coloneqq \left\{v \in A \colon {}^{2kd}/_{\varepsilon} \cdot (1+\varepsilon)^{\alpha-1} < d_j(v) \leq {}^{2kd}/_{\varepsilon} \cdot (1+\varepsilon)^{\alpha}\right\}.$$

For all the smaller degrees, we have the following bucket.

$$\mathbb{A}_\ell(j, 0) \coloneqq \left\{v \in A \colon d_j(v) \leq {}^{2kd}/_{\varepsilon}\right\}.$$

We create bags as defined in Section 7.2. Let $\mathbf{V}_\ell = \{0, 1, \ldots, \lambda\}^{r-z}$. Consider an arbitrary vector $\mathbf{v}' = (\alpha_1, \ldots, \alpha_{r-z}) \in \mathbf{V}_\ell$. Then, $\mathfrak{B}_\ell(\mathbf{v}') \coloneqq \bigcap_{j=z+1}^{r} \mathbb{A}_\ell(j, \alpha_{j-z})$. If $u, w \in \mathfrak{B}_\ell(\mathbf{v}')$, $d_j(u) \leq (1+\varepsilon) d_j(w)$, for each $j \in \mathsf{T}_{\mathsf{large}}$.

Let $\mathfrak{B}(\mathbf{v}, \mathbf{v}') = \mathfrak{B}_s(\mathbf{v}) \cap \mathfrak{B}_\ell(\mathbf{v}')$. We call any such $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$ a *bag*. The number of bags is upper bounded by $(2^{\frac{2k^2rd}{\varepsilon}} + 1)^r (\lceil \log_{(1+\varepsilon)} {}^{\varepsilon n}/_{2kd} \rceil)^r = 2^{\mathcal{O}(k^2 r^2 d/\varepsilon)} \cdot ({}^{\log n}/_{\varepsilon})^r$ via standard arguments. Note that these bags form a covering and not a partition of vertices in A as was the case in Section 7.3.

Our main idea is as follows. We start by guessing a bag that has a non-empty intersection with a feasible solution. Since any pair of vertices belonging to the same bag are adjacent to vertices with the same set of labels, any vertex in the bag can be chosen in order to satisfy the coverage requirement if colors in T_{small}. Further, the j-degree of vertices in the same bag is *almost* equal for every $j \in T_{\mathsf{large}}$. We will demonstrate that selecting a vertex v from the guessed bag leads to one of the following two possibilities:

1. either v belongs to the solution or,
2. there exists a set of vertices such that, for every vertex in this set, for some $j \in T_{\mathsf{large}}$, their j-neighborhood has a *high* overlap with v's j-neighborhood.

We utilize the definition as well as the lemma introduced in Chapter 5.2 to elaborate on the concept of *high* intersection (or overlap) with the j-neighborhood of a vertex.

Definition 11 (β_j-High Degree Set)**.** *Given a $K_{d,d}$-free bipartite graph $G = (A, B, E)$, a set $X \subseteq B$, a color $j \in [r]$, and a positive integer $\beta_j > 1$, the β_j-*High Degree Set*, denoted by $\mathsf{HD}^G_{\beta_j}(X) \subseteq A$, is a set of vertices such that every vertex $v \in \mathsf{HD}^G_{\beta_j}(X)$ satisfies $|N_j(v) \cap X| \geq \frac{|X|}{\beta_j}$, i.e.,*

$$\mathsf{HD}^G_{\beta_j}(X) = \{v \in A \colon |N_j(v) \cap X| \geq \tfrac{|X|}{\beta_j}\}$$

Let $\mathsf{AHD}^G_{\beta_j}(X) = \mathsf{HD}^G_{\beta_j}(X) \cap \{v \in A : |N_j(v)| \geq d\}$. That is, $\mathsf{AHD}^G_{\beta_j}(X)$ consists of vertices whose j-degree is at least d and its j-degree in X is at least $1/\beta_j$ fraction of X. Due to Lemma 11 in Chapter 5.2, we know that $|\mathsf{AHD}^G_{\beta_j}(X)| \leq (d-1)(2\beta_j)^d$, for all $X \subseteq B$, d, and for all $\beta_j > 1$ with $\frac{|X|}{2\beta_j} > d$.

The formal algorithmic description is presented in Algorithm 4.

Lemma 10. *Given a* yes *instance $\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t, \varepsilon)$ of* PCCDS *where G is a $K_{d,d}$-free graph, Algorithm 4 finds a set $S \subseteq V(G)$ of size at most k such that for every $j \in \mathsf{T}_{\mathsf{large}}$, $|N_j(S)| \geq (1 - k\varepsilon)t_j$, and for every $j \in \mathsf{T}_{\mathsf{small}}$, $|N_j(S)| \geq t_j$ with probability at least $e^{-2k^2 dr/\varepsilon}$.*

Proof. We prove it by induction on k.

Base Case: When $k = 0$, then we cannot cover any vertex; thus the statement holds trivially.

Induction Hypothesis: Suppose that the claim is true for $k \leq \ell - 1$.

Algorithm 4 $K_{d,d}$-FREE-PCCDS$(\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t, \varepsilon))$

1: **if** $k = 0$ **then**
2: **return** $\emptyset$ if $t_j = 0$, for every $j \in [r]$; otherwise **return** no instance.
3: **end if**
4: Compute $\mathsf{T}_{\mathsf{small}}$ and $\mathsf{T}_{\mathsf{large}}$.
5: Label the vertices in B_{small} as defined above in the green box.
6: Let $\mathcal{B}$ be the set of bags returned by applying BUCKETING(LARGE) w.r.t. $\mathsf{T}_{\mathsf{large}}$ and BUCKETING(SMALL) w.r.t. $\mathsf{T}_{\mathsf{small}}$.
7: Let $Z = \emptyset$.
8: **for** every bag $\mathfrak{B}(\mathbf{v}, \mathbf{v}') \in \mathcal{B}$ **do**
9: Select an arbitrary vertex $p \in \mathfrak{B}(\mathbf{v}, \mathbf{v}')$.
10: Compute $\mathsf{AHD}^G_{\beta_j}(N_j(p))$ with $\beta_j = \frac{k}{\varepsilon}$ for each $j \in \mathsf{T}_{\mathsf{large}}$.
11: Let $Z = Z \cup \left(\bigcup_{j \in \mathsf{T}_{\mathsf{large}}} \mathsf{AHD}^G_{\beta_j}(N_j(p)) \cup \{p\} \right)$.
12: **end for**
13: **for** each $y \in Z$ **do**
14: let $\mathcal{I}_y \leftarrow$ PRUNEINSTANCE$(\mathcal{I}, y)$
15: let S_y be the set returned by $K_{d,d}$-FREE-PCCDS$(\mathcal{I}_y)$
16: **end for**
17: Among all the sets S_y, $y \in Z$, suppose z is the element such that for every $j \in [r]$, $|N_j(\{z\} \cup S_z)| \geq (1 - k\varepsilon)t_j$. Return $\{z\} \cup S_z$. Return no instance if no such z exists.

Inductive Step: Next, we prove the claim for $k = \ell$. Let $S^\star \subseteq A$ such that $|S^\star| = \ell$ and for every $j \in [r]$, $|N_j(S^\star)| = \widetilde{t}_j \geq t_j$. We consider the following two cases.

Case 1. $S^\star \cap Z = \emptyset$.

Case 1.1. Suppose that $\mathsf{T}_{\mathsf{small}} = [r]$, i.e., for all $j \in [r]$, $t_j \leq 2\ell^2 d/\varepsilon$.

In this case, $N(S^\star)$ is colorful with probability at least $e^{-2\ell^2 dr/\varepsilon}$. Thus, $S^\star$ has non-empty intersection with at least one bag of the type $\mathfrak{B}(\mathbf{v}, \mathbf{0})$ with probability at least $e^{-2\ell^2 dr/\varepsilon}$. Here, $\mathbf{v} = (\gamma_1, \ldots, \gamma_r)$ with $\gamma_j \subseteq \mathsf{labels}$ for each $j \in [r]$, and $\mathbf{0}$ is the zero vector, i.e., the unique vector of length 0.

Let $x \in S^\star \cap \mathfrak{B}(\mathbf{v}, \mathbf{0})$ and let $S^\star_x = S^\star \setminus \{x\}$. Clearly, for every $j \in [r]$, $|N^G_j(S^\star_x)| \geq t_j - |\mathsf{lab}(N^G_j(x))| = t_j - |\gamma_j|$. Let p be an arbitrary vertex in $\mathfrak{B}(\mathbf{v}, \mathbf{0})$ selected in Step 10 of the algorithm. Due to the construction of the bucket $\mathfrak{B}(\mathbf{v}, \mathbf{0})$, we know that $\mathsf{lab}(N^G_j(p)) = \mathsf{lab}(N^G_j(x)) = \gamma_j$, for each $j \in [r]$. Due to induction hypothesis, Algorithm 4 finds a set $S_p \subseteq A \setminus \{p\}$ of size $\ell - 1$ such that $|N^G_j(S_p)| \geq t_j - |\gamma_j|$ with probability at least $e^{-2(\ell-1)^2 dr/\varepsilon}$, for every $j \in [r]$. Hence, for every $j \in [r]$, $|N^G_j(\{p\} \cup S_p)| \geq t_j$

with probability at least $e^{-2(\ell-1)^2 dr/\varepsilon}$ which is at least $e^{-2\ell^2 dr/\varepsilon}$ for $\ell \geq 1/2$. Since ℓ is an integer, this holds for any $\ell \geq 1$.

Case 1.2. Next, we consider the case when $\mathsf{T}_{\mathsf{large}} \neq \emptyset$.

Let $\mathfrak{B}(\mathbf{v}, \mathbf{v}') \in \mathcal{B}$ be the bag with which $S^\star$ has non-empty intersection. Let $x \in S^\star \cap \mathfrak{B}(\mathbf{v}, \mathbf{v}')$ and let $S^\star_x = S^\star \setminus \{x\}$. Clearly, for every $j \in [r]$, $|N^G_j(S^\star_x)| \geq t_j - d^G_j(x)$. Let p be an arbitrary vertex in $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$ selected in Step 10 of the algorithm. Since $S^\star \cap Z = \emptyset$, for every $j \in \mathsf{T}_{\mathsf{large}}$ and for every $w \in S^\star$, $|N^G_j(w) \cap N^G_j(p)| < {}^{|N^G_j(p)|}/_{\beta_j}$ holds. Thus, for every $j \in \mathsf{T}_{\mathsf{large}}$,

$$\begin{aligned}
|N^{G_p}_j(S^\star_x)| &= |N^G_j(S^\star_x) \setminus N^G_j(p)| \\
&\geq |N^G_j(S^\star_x)| - |N^G_j(p) \cap N^G_j(S^\star_x)| \\
&\geq t_j - d^G_j(x) - \ell \frac{|N^G_j(p)|}{\beta_j} \\
&\geq t_j - d^G_j(x) - \varepsilon |N^G_j(p)|
\end{aligned}$$

Here, G_p is the graph obtained after removing p and its neighbors from G, i.e., $G_p = (A \setminus \{p\}, B \setminus N(p))$. Due to induction hypothesis, Algorithm 4 finds a set $S_p \subseteq A \setminus \{p\}$ of size $\ell - 1$ such that, for every $j \in \mathsf{T}_{\mathsf{large}}$,

$$\begin{aligned}
|N^{G_p}_j(S_p)| &\geq (1 - (\ell-1)\varepsilon)|N^{G_p}_j(S^\star_x)| \\
&\geq (1 - (\ell-1)\varepsilon)(t_j - d^G_j(x) - \varepsilon |N^G_j(p)|).
\end{aligned}$$

Since p and x belong to the same bag, we have $d^G_j(x) \leq (1+\varepsilon) d^G_j(p)$. Thus,

$$\begin{aligned}
|N^G_j(\{p\} \cup S_p)| &= |N^G_j(p)| + |N^{G_p}_j(S_p)| \\
&\geq d^G_j(p) + (1 - (\ell-1)\varepsilon)(t_j - d^G_j(x) - \varepsilon d^G_j(p)) \\
&\geq d^G_j(p) + (1 - (\ell-1)\varepsilon)(t_j - (1+\varepsilon) d^G_j(p) - \varepsilon d^G_j(p)) \\
&\geq (1 - \ell\varepsilon) t_j + \varepsilon t_j + d^G_j(p)((\ell-3)\varepsilon + 2(\ell-1)\varepsilon^2) \\
&\overset{(☺)}{\geq} (1 - \ell\varepsilon) t_j
\end{aligned}$$

(☺) holds for any $\ell \geq 2$ and using the fact that $d^G_j(p) \leq t_j$. When $\ell = 1$, $k \leq 0$ and this is handled by the base case.

We next argue the claim for $j \in \mathsf{T}_{\mathsf{small}}$. Using the same argument as above (when we considered $\mathsf{T}_{\mathsf{small}} = [r]$ in Case 1.1), we have that for every $j \in \mathsf{T}_{\mathsf{small}}$, $|N^G_j(\{p\} \cup S_p)| \geq t_j$ with probability at least $e^{-2\ell^2 dr/\varepsilon}$.

Case 2. $S^\star \cap Z \neq \emptyset$.

Let $x \in S^\star \cap Z$ and let $S^\star_x = S^\star \setminus \{x\}$. For every $j \in [r]$, clearly, $|N^G_j(S^\star_x)| \geq t_j - d^G_j(x)$.

Due to induction hypothesis, we know that Algorithm 4 finds a set $S_x \subseteq A \setminus \{x\}$ of size $\ell - 1$ such that, for every $j \in \mathsf{T}_{\mathsf{large}}$, $|N^{G_x}_j(S_x)| \geq (1-(\ell-1)\varepsilon)(t_j - d^G_j(x))$, and, for every $j \in \mathsf{T}_{\mathsf{small}}$, $|N^{G_x}_j(S_x)| \geq t_j - |\mathsf{lab}(N^G_j(x))|$ with probability at least $e^{-2(\ell-1)^2 dr/\varepsilon}$. Here, G_x is the graph obtained after removing x and its neighbors from G, i.e., $G_x = (A \setminus \{x\}, B \setminus N(x))$.

Thus, for every $j \in \mathsf{T}_{\mathsf{large}}$,

$$\begin{aligned} |N^G_j(\{x\} \cup S_x)| &= |N^G_j(x)| + |N^{G_x}_j(S_x)| \\ &\geq d^G_j(x) + (1-(\ell-1)\varepsilon)(t_j - d^G_j(x)) \\ &\geq (1-\ell\varepsilon)t_j \end{aligned}$$

For every $j \in \mathsf{T}_{\mathsf{small}}$, the argument is similar to that in Case 1.

□

Thus, we obtain the following result by invoking Algorithm 4 with $\varepsilon' = \varepsilon/k$.

Theorem 7. *There exists a randomized algorithm that runs in* $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} n^{\mathcal{O}(1)}$ *time, and given a* yes *instance* $(\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t, \varepsilon))$ *of* PCCDS *where* G *is a* $K_{d,d}$*-free graph, finds a set* $S \subseteq V(G)$ *of size at most* k *such that for every* $j \in \mathsf{T}_{\mathsf{large}}$, $|N_j(S)| \geq (1-\varepsilon)t_j$, *and for every* $j \in \mathsf{T}_{\mathsf{small}}$, $|N_j(S)| \geq t_j$ *with probability at least* $e^{-2k^2 dr/\varepsilon}$.

Proof. We call Algorithm 4 with $\varepsilon' = \varepsilon/k$. The correctness follows due to Lemma 10. Recall that the number of bags is upper bounded by $2^{\mathcal{O}(k^2 r^2 d/\varepsilon')} \cdot (\log n/\varepsilon')^r$. Furthermore, for every $x \in \mathfrak{B}(\mathbf{v}, \mathbf{v}')$, $|\bigcup_{j \in \mathsf{T}_{\mathsf{large}}} \mathsf{AHD}^G_{\beta_j}(N_j(x))| \leq r(d-1)(2\beta_j)^d$. Thus, in every recursive call,

$$|Z| \leq 2^{\mathcal{O}(k^3 r^2 d/\varepsilon)} \cdot (k \log n/\varepsilon)^r \cdot (1 + r(d-1)(2k^2/\varepsilon)^d) = 2^{\mathcal{O}(k^3 r^2 d/\varepsilon)} \cdot (\log n)^{\mathcal{O}(r)}.$$

Since the number of recursive levels is bounded by k, the running time is $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} n^{\mathcal{O}(1)}$. □

We derandomize this algorithm using (p, q)-perfect hash family to obtain a deterministic algorithm in the following theorem.

Theorem 8. *There exists a deterministic algorithm that runs in* $2^{\mathcal{O}(\frac{k^4r^2d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)}n^{\mathcal{O}(1)}$ *time, and given a* yes *instance* $(\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t, \varepsilon))$ *of* PCCDS *where* G *is a* $K_{d,d}$*-free graph, finds a set* $S \subseteq A$ *of size at most* k *such that, for every* $j \in [r]$, $|N_j(S)| \geq (1-\varepsilon)t_j$.

Proof of Theorem 8. We use the notion of (p, q)-perfect hash family[3] for the notion of derandomization. Let $\mathcal{I}$ be an instance of PCCDS. Instead of taking a random coloring for B_{small} in Algorithm 4, we create a $(|B_{\mathsf{small}}|, \frac{2k^2rd}{\varepsilon})$-perfect hash family $\mathcal{F}$, and run the algorithm for every label function $f \in \mathcal{F}$. Using this, we get the proof of Theorem 8. □

We now state our result for F-Max k-weight SAT on $K_{d,d}$-free formulae. This is obtained by first applying the randomized reduction in Chapter 6 followed by the EPAS in Theorem 8.

Corollary 2. *Let* $\varepsilon > 0$. *Then,* F-Max k-weight SAT *admits a randomized* EPAS *with running time* $2^{\mathcal{O}(\frac{k^4r^2d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)}(n+m)^{\mathcal{O}(1)}$ *on* $K_{d,d}$*-free formulas. This result holds with constant probability.*

[3]The reader can refer to Definiton 5 and Proposition 3 in Chapter 4 for details.

Chapter 8

Fairness and Matroid Constrained Partial Coverage and Satisfiability

This chapter presents the same core results in the previous chapter, namely, EPAS parameterized by the solution size for F-Max k-weight SAT and F-Maximum Coverage, now extended to incorporate additional matroid constraints. To handle this, we leverage the concept of representative sets in a careful and non-trivial way, allowing us to preserve both feasibility under the matroid and the desired approximation guarantees. For the ease of exposition, we recast our problem (M, F)-Maximum Coverage to PMCCDS.

Partition Matroid & Color Constrained Dominating Set (PMCCDS)
Input: An instance $\mathcal{I} = (G, r, f, \mathcal{M}, k, t)$, where $G = (A \uplus B, E)$ is a bipartite graph, $f : B \to [r]$ is a surjective function, a matroid $\mathcal{M} = (A, I)$ defined on A, a non-negative integer k, and for each color $j \in [r]$, a *coverage requirement* $t_j \geq 0$.
Here, $B_j \coloneqq \{\mathfrak{s} \in B : f(\mathfrak{s}) = j\}$ is a set of vertices of *color* j, for each $j \in [r]$.
Parameter: k
Question: Does there exist a subset $S \subseteq A$, such that (1) $|S| \leq k$, (2) $S \in I$, and (3) for each $j \in [r]$, $|N_j(S)| \geq t(j)$? Here, $N_j(S) \subseteq B_j$ is the set of vertices of color $j \in [r]$ that are adjacent to at least one vertex in S.

Observe that finding a subfamily of size k that is independent in the given matroid and covers at least t_j elements of the color j is equivalent to finding a set $S \subseteq A$ of size k such that S is an independent set in the given $\mathcal{M}$ of rank at most k, and $N_j(S)$

(neighbors of S that are colored j) is at least t_j in the incidence graph. Without loss of generality, we assume that k equals the rank of $\mathcal{M}$, i.e., the solution is a base of $\mathcal{M}$ by truncating the matroid appropriately. Note that it is straightforward to work with the truncated matroid, given oracle access to the original matroid.

By slightly abusing the notation, we use $\mathcal{M}$ to denote the appropriately truncated matroid, if necessary.

We give an EPAS for PMCCDS on $K_{d,d}$-free graphs in Section 8.2. Finally, in Section 8.3, we design and analyze a more efficient EPAS for PMCCDS when each vertex in B has degree at most d (i.e., frequency at most d).

8.1 Handling Matroid Constraints: Overview of the technical idea

First, we consider M-Maximum Coverage (note that this is an orthogonal generalization of Maximum Coverage, without multiple coverage constraints), where the solution is required to be an independent set in the given matroid $\mathcal{M} = (\mathcal{F}, I)$ of rank k. We assume that we are given an *oracle access* to $\mathcal{M}$ in the form of an algorithm that answers the queries of the form "Is $\mathcal{R}$ an independent set?" for any subset $\mathcal{R} \subseteq \mathcal{F}$. Let us revisit the initial deterministic EPAS for Maximum Coverage and try to generalize it to M-Maximum Coverage. Recall that this algorithm branches on each set $S \in \mathcal{H}(L)$, where L is a largest set. The analysis of easy case goes through even in presence of the matroid constraint, since we branch on a set from an optimal solution $\mathcal{O}$. However, in the hard case, our analysis replaces a $S_l \in \mathcal{O}$ with L, and argues that the branch corresponding to L returns a $(1-\varepsilon)$-approximate solution. However, this does not work for M-Maximum Coverage, since $(\mathcal{O} \setminus \{S_l\}) \cup \{L\}$ may not be an independent set in $\mathcal{M}$. To summarize, although L handles the coverage constraints (approximately), it may fail to handle the matroid constraint. In fact, it may just so happen that L is not a good set *at all*, in the sense that, for *any* set $S \in \mathcal{O}$, $(\mathcal{O} \setminus \{S\}) \cup \{L\}$ is not independent in $\mathcal{M}$, which is crucial for the induction to go through.

To solve this issue, we resort to the bucketing idea as in the fair coverage case (thus, the subsequent arguments generalize to (M, F)-Maximum Coverage in a straightforward manner; although let us stick to the special case of M-Maximum Coverage for now). Indeed, branching w.r.t. the largest set L is an overkill –

it suffices to pin down a bag $\mathcal{B}$ containing a set in $\mathcal{O}$ (it does not even have to be the largest set), by guessing from $\mathcal{O}\left(\frac{\log k}{\varepsilon}\right)$ bags. However, we again cannot select an arbitrary set $S \in \mathcal{B}$ and define heavy sets w.r.t. S, precisely due to the matroid compatibility issues mentioned earlier. Therefore, we resort to the idea of *representative sets* from matroid theory [36] [1] Assuming our guess for $\mathcal{B}$ is correct, there exists some $S \in \mathcal{B} \cap \mathcal{O}$. However, we cannot further "guess" S, since the size of the bag may be too large. Instead, we compute a inclusion-wise maximal independent set $\mathcal{B}' \subseteq \mathcal{B}$. Note that the size of $\mathcal{B}'$ is at most k, and it can be computed using polynomially many queries to the independence oracle. However, it may very well happen that $S \notin \mathcal{B}'$. Nevertheless, using matroid properties, we can argue that, there exists a set $S' \in \mathcal{B}'$, such that $(\mathcal{O} \setminus \{S\}) \cup \{S'\}$ is an independent set. Thus, $\mathcal{B}'$ is a *representative set* of $\mathcal{B}$. Furthermore, since both S and S' come from the same bag, they cover approximately the same number of elements. Thus, our modified deterministic algorithm works as follows. First, it computes maximal independent set $\mathcal{B}' \subseteq \mathcal{B}$, and for each $S' \in \mathcal{B}'$, it computes the heavy family $\mathcal{H}(S')$. Then, it branches over all sets in $\bigcup_{S' \in \mathcal{B}'} \mathcal{H}(S')$. If one of the branches corresponds to branching on a set from $\mathcal{O}$, then the analysis is similar to the easy case. Otherwise, we know that $(\mathcal{O} \setminus \{S\}) \cup \{S'\}$ is an $(1-\varepsilon)$-approximate solution that is also an independent set. Therefore, the branch corresponding to S' yields the required $(1-\varepsilon)$-approximation. We can improve the running time via doing a randomized branching in two steps: first we pick a set $S'' \in \mathcal{B}'$ uniformly at random, and then we perform the probabilistic branching using the weights $h_{S''}(\cdot)$.

Both deterministic and randomized variants incur a further multiplicative overhead of $(\frac{k \log k}{\varepsilon})^k$ due to first guessing a bag, and then computing a representative set $\mathcal{B}' \subseteq \mathcal{B}$ of the bag, and thus do not improve over the results of Sellier [31] in terms of running time for M-Maximum Coverage. However, this idea naturally generalizes to (M, F)-Maximum Coverage, with the appropriate modifications in bucketing (as mentioned in the previous paragraph) to handle the multiple coverage requirements of different colors.

[1] Although this is a powerful hammer in its full generality—which we do use to handle multiple matroid constraints—our specialized setting lets us use much simpler arguments to handle single matroid constraint in M-Maximum Coverage/(M, F)-Maximum Coverage.

8.2 $K_{d,d}$-free Case

We design an EPAS for PMCCDS on $K_{d,d}$-free graphs. The algorithm is largely similar to the one in Section 7.4, with a few modifications as described next.

In addition to the standard inputs for Algorithm 4, the modified algorithm instance also receives (oracle access to) a matroid $\mathcal{M}' = (A', I')$. Here, $\mathcal{M}'$ is obtained by contracting the original matroid $\mathcal{M}$ on the set of elements Q added so far leading to this recursive call. From the definition of matroid contraction, it follows that any independent set in $\mathcal{M}'$, along with Q, is independent in the original matroid $\mathcal{M}$. Due to our initial truncation, we can inductively assume that $\mathcal{M}'$ has rank exactly k, where $0 \leq k \leq k^\star$, where $k^\star$ is the *original* budget (and thus the rank of the original matroid $\mathcal{M}$). Furthermore, rather than just searching for a set of size k that satisfies the coverage requirements for each color class, the algorithm seeks a set S that meets two conditions: first, it must be independent in $\mathcal{M}'$, and second, it must have a neighborhood size $N_j(S) \geq (1-\varepsilon)t_j$, thus satisfying the coverage requirements for each color class $j \in [r]$.

Similar to Algorithm 4, we start by guessing a bag $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$ that contains a vertex of a feasible solution $S^\star$ (i.e., we branch on all such bags). Here, $S^\star \subseteq A$, $S^\star \in I'$, $|S^\star| \leq k$ and $|N_j(S^\star)| \geq t_j$, for each $j \in [r]$. But instead of selecting any arbitrary vertex in $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$ (as done in line 5 of Algorithm 4), we compute a $R(\mathfrak{B}(\mathbf{v}, \mathbf{v}')) \subseteq^{k-1}_{\text{rep}} \mathfrak{B}(\mathbf{v}, \mathbf{v}')$. Lemma 1 implies that, $|R(\mathfrak{B}(\mathbf{v}, \mathbf{v}'))| \leq k$, and it can be computed in polynomial time. Next, for every $v_i \in R(\mathfrak{B}(\mathbf{v}, \mathbf{v}'))$, we compute the sets $\mathsf{AHD}^G_{\beta_j}(N_j(v_i))$. Next, we define

$$Z_{\mathbf{v},\mathbf{v}'} \coloneqq \bigcup_{v_i \in R(\mathfrak{B}(\mathbf{v},\mathbf{v}'))} \bigcup_{j \in \mathsf{T}_{\mathsf{large}}} \{v_i\} \cup \mathsf{AHD}^G_{\beta_j}(N_j(v_i)) \tag{8.1}$$

We branch on such a $y \in Z_{\mathbf{v},\mathbf{v}'}$ and update the instance $\mathcal{I}'$ passed to the next iteration of the algorithm as was done in the PruneInstance procedure, with the following modification. First, we obtain $\mathcal{M}'' \coloneqq \mathcal{M}'/y$, i.e., $\mathcal{M}''$ is obtained by contracting $\mathcal{M}'$ on the vertex y on which we are branching. Note that one can simulate oracle access to $\mathcal{M}''$ using the oracle access to $\mathcal{M}'$ by always including y in the set being queried. Note that the rank of $\mathcal{M}''$ is $k-1$.

The formal algorithmic description is presented in Algorithm 5.

Lemma 11. *Given a* yes *instance* $(\mathcal{I} = (G = (A \uplus B, E), [r], f, t, k, \varepsilon), \mathcal{M} = (A, I))$ *of* PMCCDS *where* G *is a* $K_{d,d}$*-free graph, Algorithm 5 finds a set* $S \subseteq A$ *of size at most* k *such that for every* $j \in \mathsf{T}_{\mathsf{large}}$, $|N_j(S)| \geq (1-k\varepsilon)t_j$, *and for every* $j \in \mathsf{T}_{\mathsf{small}}$,

Algorithm 5 $K_{d,d}$-FREE-PMCCDS$(\mathcal{I} = (G = (A \uplus B, E), [r], f, t, k, \varepsilon), \mathcal{M})$

if $k = 0$ **then**
  return $\emptyset$ if $t_j = 0$, for every $j \in [r]$; otherwise **return** no instance.
end if
Compute $\mathsf{T}_{\mathsf{small}}$ and $\mathsf{T}_{\mathsf{large}}$.
Label the vertices in B_{small} as defined above in the green box.
Let $\mathcal{B}$ be the set of bags returned by applying BUCKETING(LARGE) w.r.t. $\mathsf{T}_{\mathsf{large}}$ and BUCKETING(SMALL) w.r.t. $\mathsf{T}_{\mathsf{small}}$.
Let $Z = \emptyset$.
for every bag $\mathfrak{B}(\mathbf{v}, \mathbf{v}') \in \mathcal{B}$ **do**
  Compute $R(\mathfrak{B}(\mathbf{v}, \mathbf{v}')) \subseteq^{k-1}_{\text{rep}} \mathfrak{B}(\mathbf{v}, \mathbf{v}')$ using Lemma 1.
  for every vertex $p \in R(\mathfrak{B}(\mathbf{v}, \mathbf{v}'))$ **do**
    Compute $\mathsf{AHD}^G_{\beta_j}(N_j(p))$ with $\beta_j = \frac{k}{\varepsilon}$ for each $j \in \mathsf{T}_{\mathsf{large}}$.
    Let $Z = Z \cup \left(\bigcup_{j \in \mathsf{T}_{\mathsf{large}}} \mathsf{AHD}^G_{\beta_j}(N_j(p)) \cup \{p\} \right)$.
  end for
end for
for each $y \in Z$ **do**
  let $\mathcal{I}_y \leftarrow$ PRUNEINSTANCEMATROID$(\mathcal{I}, y)$
  let S_y be the set returned by $K_{d,d}$-FREE-PMCCDS$(\mathcal{I}_y)$
end for
Among all the sets S_y, $y \in Z$, suppose z is the element such that for every $j \in [r]$, $|N_j(\{z\} \cup S_z)| \geq (1 - k\varepsilon)t_j$. Return $\{z\} \cup S_z$. Return no instance if no such z exists.

$|N_j(S)| \geq t_j$ with probability at least $e^{-2k^2 dr/\varepsilon}$. Moreover, the set S is an independent set in $\mathcal{M}$.

Proof. We prove it by induction on k.

Base Case: When $k = 0$, then we cannot cover any vertex; thus the statement holds trivially.

Induction Hypothesis: Suppose that the claim is true for $k \leq \ell - 1$.

Inductive Step: Next, we prove the claim for $k = \ell$. Let $S^\star \subseteq A$ such that $|S^\star| = \ell$, $S^\star \in I$ and for every $j \in [r]$, $|N_j(S^\star)| = \widetilde{t_j} \geq t_j$. We consider the following two cases.

Case 1. $S^\star \cap Z = \emptyset$.

Recall that $\mathfrak{B}(\mathbf{v}, \mathbf{v}') \cap S^\star \neq \emptyset$, i.e., $S^\star$ selects at least one vertex, say x from $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$. However, $S^\star \cap Z = \emptyset$, which implies that $x \notin Z$, which, in particular,

implies that $x \notin R(\mathfrak{B}(\mathbf{v}, \mathbf{v}'))$. In this case, based on the correctness arguments of Algorithm 4, we know that any vertex $y \in \mathfrak{B}(\mathbf{v}, \mathbf{v}') \cap Z$ serves as a suitable "approximate replacement" for x, as far as the coverage requirement is concerned.

However, here we have an additional requirement that that the solution be an independent set in $\mathcal{M}$. To this end, let $S^\star_x = S^\star \setminus \{x\}$. Note that $|S^\star_x| = k-1$, and $S^\star_x \cup \{x\}$ is independent in $\mathcal{M}$. It follows that, there exists some $y \in R(\mathfrak{B}(\mathbf{v}, \mathbf{v}')) \subseteq^{k-1}_{\text{rep}} \mathfrak{B}(\mathbf{v}, \mathbf{v}')$ such that $\{y\} \cap S^\star_x = \emptyset$ and $\{y\} \cup S^\star_x$ also independent in $\mathcal{M}$. Thus, using inductive hypothesis, the solution returned by the recursive call corresponding to y, combined with y, is (1) independent in $\mathcal{M}$, and (2) satisfies the coverage requirements up to an $1-\varepsilon$ factor.

Case 2. $S^\star \cap Z \neq \emptyset$.

Let $x \in Z \cap S^\star$. By induction our algorithm on the instance returned by PruneInstanceMatroid $(\mathcal{I}, x)$ with the contracted matroid $\mathcal{M}'' = \mathcal{M}/x$, returns a set of size S' of size $k-1$ that satisfies the approximate coverage requirements (as argued for Algorithm 4). By the definition of $\mathcal{M}''$, it follows that $S' \cup \{x\}$ is independent in $\mathcal{M}$.

□

Thus, we obtain the following result by invoking Algorithm 5 with $\varepsilon' = \varepsilon/k$.

Theorem 9. *There exists a randomized algorithm that runs in* $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} n^{\mathcal{O}(1)}$ *time, and given a* yes *instance* $(\mathcal{I} = (G = (A \uplus B, E), [r], f, t, k, \varepsilon), \mathcal{M})$ *of* PMC-CDS *where* G *is a* $K_{d,d}$*-free graph, finds a set* $S \subseteq V(G)$ *of size at most* k *such that* S *is an independent set in* $\mathcal{M}$*, for every* $j \in \mathsf{T}_{\mathsf{large}}$*,* $|N_j(S)| \geq (1-\varepsilon)t_j$*, and for every* $j \in \mathsf{T}_{\mathsf{small}}$*,* $|N_j(S)| \geq t_j$ *with probability at least* $e^{-2k^2 dr/\varepsilon}$*.*

Proof. We call Algorithm 5 with $\varepsilon' = \varepsilon/k$. The correctness follows due to Lemma 11.

Recall that the number of bags is upper bounded by $2^{\mathcal{O}(k^2 r^2 d/\varepsilon')} \cdot (\log n/\varepsilon')^r$. We compute the set of high neighbors with respect to all colors in $\mathsf{T}_{\mathsf{large}}$, i.e., $\bigcup_{j \in \mathsf{T}_{\mathsf{large}}} \mathsf{AHD}^G_{\beta_j}(N_j(x))$, for every vertex x in $R(\mathfrak{B}(\mathbf{v}, \mathbf{v}'))$ for a bag $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$. Thus, the branching factor increases by at most k. This adds a multiplicative factor of k^k to the running time, which is absorbed into the FPT factor. Furthermore, the time required to compute a representative set $R(\mathfrak{B}(\mathbf{v}, \mathbf{v}'))$ and the set $Z_{\mathbf{v},\mathbf{v}'}$ being polynomial-time for any bag $\mathfrak{B}(\mathbf{v}, \mathbf{v}')$ is absorbed into the polynomial factor. □

We derandomize this algorithm using (p, q)-perfect hash family to obtain a deterministic algorithm in the following theorem.

Theorem 10. *There exists a deterministic algorithm that runs in*

$$2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} n^{\mathcal{O}(1)}$$

time, and given a yes *instance* $(\mathcal{I} = (G = (A \uplus B, E), [r], f, k, t, \varepsilon))$ *of* PMCCDS *where* G *is a* $K_{d,d}$*-free graph, finds a set* $S \subseteq A$ *of size at most* k *such that,* S *is independent in* $\mathcal{M}$*, for every* $j \in \mathsf{T}_{\mathsf{large}}$*,* $|N_j(S)| \geq (1-\varepsilon)t_j$*, and for every* $j \in \mathsf{T}_{\mathsf{small}}$*,* $|N_j(S)| \geq t_j$*.*

We now state our result for (M, F)-Max k-weight SAT on $K_{d,d}$-free formulae. This is obtained by first apply the randomized reduction in Chapter 6 followed by the EPAS in Theorem 9.

Corollary 3. *Let* $\varepsilon > 0$*. Then,* (M, F)-Max k-weight SAT *admits a randomized* EPAS *with running time* $2^{\mathcal{O}(\frac{k^4 r^2 d}{\varepsilon})}(\log n)^{\mathcal{O}(rk)} \cdot (n+m)^{\mathcal{O}(1)}$ *on* $K_{d,d}$*-free formulas. This result holds with constant probability.*

8.3 Frequency d Case

Our algorithm for PCCDS, when each vertex in B has degree at most d is given in Algorithm 6. In addition to the standard inputs for Algorithm 3, the modified algorithm instance is provided a matroid $\mathcal{M}$ as input. We now seek an independent set in $\mathcal{M}$ that satisfies the coverage requirements approximately. Similar to Algorithm 3, we start by guessing a bag $A(\mathbf{v})$ that contains a vertex of an optimal solution S. A solution S is called optimal if it is independent in $\mathcal{M}$ and satisfies the requirements for every color class. But instead of selecting any arbitrary vertex in $A(\mathbf{v})$ (as done in Step 5 of Algorithm 3), we compute a representative set $R(A(\mathbf{v})) \subseteq^{k-1}_{\text{rep}} A(\mathbf{v})$ of size at most k in polynomial-time. We choose a vertex v uniformly at random from this set and subsequently proceed in accordance with the steps outlined in Algorithm 3. We contend that with a *high* probability, either vertex v or a vertex w (based on the probability distribution $p(w)$) can be included in the gradually constructed solution (produced by an iteration of the same algorithm with smaller k) without compromising independence, while still satisfying the approximate coverage requirements. Moreover, the PruneInstance procedure undergoes identical modifications as detailed in the preceding section.

Algorithm 6 PMCCDS($\mathcal{I} = (G = (A \uplus B, E), [r], f, \mathcal{M}, k, t)$)

1: **if** $k = 0$ **then**
2: **return** $\emptyset$
3: **end if**
4: Let $\mathcal{A}$ be the set of bags returned by applying BUCKETING on $\mathcal{I}$
5: Choose a bag $A(\mathbf{v}) \in \mathcal{A}$ uniformly at random, and compute the representative set $R(A(\mathbf{v})) \subseteq^{k-1}_{\text{rep}} A(\mathbf{v})$.
6: Select an arbitrary vertex $v \in R(A(\mathbf{v}))$.
7: Define the following quantities w.r.t. the vertex v:
8: • For any $w \in A \setminus v$ and any color j, let $h_j(w) \coloneqq \frac{|N^G_j(w) \cap N^G_j(v)|}{|N^G_j(v)|}$,
9: • Let $p(w) \coloneqq \frac{1}{2rd} \cdot \sum_{j=1}^{r} h_j(w)$
10: • Let $p(v) \coloneqq \frac{1}{2}$
11: Sample a vertex $u \in V(G)$ at random proportional to the quantities $p(\cdot)$
12: $\mathcal{I}' \leftarrow$ PRUNEINSTANCEMATROID$(\mathcal{I}, u)$ ▷ Residual instance after adding u to the solution
13: **return** $\widetilde{S} \cup \{u\}$, where $\widetilde{S} \leftarrow$ PMCCDS$(\mathcal{I}')$

We establish the correctness of the algorithm below.

Notice that by Observation 2, the values calculated in steps 9 and 10 of Algorithm 6 form a valid probability distribution. That is, the sum of $p(v)$ over all $v \in A$ is at most 1.

Lemma 12. *Consider a recursive call* PMCCDS $(\mathcal{I})$, *where* $\mathcal{I} = (G = (A \uplus B, E), [r], f, \mathcal{M}, k, t)$, *where* $\mathcal{M} = (A, I)$ *is a matroid, and let* k^* *be the value of* k *from the* original instance. *Consider a set* $S \subseteq A$ *of size* k *such that* $S \in I$, *and for each* $j \in [r]$, $|N^G_j(S)| = \widetilde{t}_j \geq t_j$.

Then, with probability at least $\left(\frac{1}{L} \cdot \frac{\varepsilon}{2rdk}\right)^k$, *the algorithm returns a subset* $S' \subseteq A$ *of size at most* k, *such that* $S' \in I$, *and for each* $j \in [r]$, $|N^G_j(S')| = t''_j$, *where*

$$t''_j \geq (1 - 2\varepsilon) \min(\widetilde{t}_j, t_j) - \frac{\varepsilon k}{k^*} \cdot t_j.$$

Proof. We prove this by induction. When $k = 0$, then there is nothing to prove.

Now suppose the claim is true for $k - 1 \geq 0$ and we prove it for k.

Fix a set $S \subseteq V(G)$ of size k such that $S \in I$, and $|N^G_j(S)| = \widetilde{t}_j \geq t_j$ for each $j \in [r]$. There exists a bag $A(\mathbf{v})$ such that $S \cap A(\mathbf{v}) \neq \emptyset$. Since in the first step (Step 5), the algorithm picks a bag uniformly at random from at most L bags, the probability that $A(\mathbf{v})$ is picked is at least $\frac{1}{L}$. We condition on the event that this choice is made. The algorithm then computes a representative set $R(A(\mathbf{v})) \subseteq^{k-1}_{\text{rep}} A(\mathbf{v})$ of size

procedure PRUNEINSTANCEMATROID($\mathcal{I}, u$)
where $\mathcal{I} = (G = (A \uplus B, E), [r], f, \mathcal{M}, k, t)$ and $u \in A$

1: Let $J \coloneqq \{j \in [r] : |N_j(u)| \geq t_j\}$ ▷ J is the set of colors already satisfied by u
2: **if** $J = \emptyset$ **then**
3: $G' \coloneqq G \setminus\setminus u$ ▷ Remove u from A and all of its neighbors from B
4: f' is the restriction of f to $B \setminus \bigcup_{j \in [r]} N_j(u)$
5: $\mathcal{M}' = \mathcal{M}/u$
6: $t'_j = t_j - |N_j(u)|$ for all colors j ▷ Account for addition of u into the solution
7: $\mathcal{I}' = (G', [r], f', \mathcal{M}', k-1, t')$
8: **else**
9: $r' \coloneqq r - |J|$, and w.l.o.g. assume that $J = \{r'+1, r'+2, \ldots, r\}$
10: $G' = G[A' \uplus B']$, where $A' \coloneqq A \setminus \{u\}$, and $B' \coloneqq B \setminus \left(N(u) \cup \bigcup_{j \in J} f^{-1}(j)\right)$
▷ Remove u, its neighbors in B, and vertices of all the colors satisfied by u
11: f' is the restriction of f to B', and $t'_j = t_j - |N_j(u)|$ for $j \in [r']$
12: $\mathcal{M}' = \mathcal{M}/u$
13: $\mathcal{I}' = (G', [r'], f', \mathcal{M}', k-1, t')$
14: **end if**
15: **return** $\mathcal{I}'$

at most k and picks an element from $R(A(\mathbf{v}))$ uniformly at random. Suppose the algorithm picks $v \in R(A(\mathbf{v}))$ in Step 6. This happens with probability at least $\frac{1}{k}$.

Suppose the set J, as defined with respect to a sampled vertex u, in Step 1 of PRUNEINSTANCEMATROID is non-empty. Then, for any $j \in J$, Observation 3 implies that $|N_j^G(u)| \geq t_j$, which is at least the bound in the lemma. Thus, it suffices to focus on colors in $[r] \setminus J$. Let G' be the graph in the instance $\mathcal{I}'$ obtained in Step 12 of Algorithm 6.

Now we consider the following different cases.

Case 1. $v \in S$.
In this case, the probability that the chosen vertex u in Step 11 is equal to v, is at least $1/2$. We condition on this event.

By Observation 3 (item 3), the set $S \setminus \{v\}$ of size $k-1$ is such that for each $j \notin J$, $|N_j^{G'}(S \setminus \{v\})| = \widetilde{t}_j - |N_j^G(v)|$. Thus, by induction hypothesis (i.e., using Lemma 9), with probability at least $\left(\frac{1}{L(k-1)} \cdot \frac{\varepsilon}{2rd}\right)^{k-1} \geq \left(\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd}\right)^{k-1}$, PCCDS($\mathcal{I}'$) returns a set $\widetilde{S}$ of size at most $k-1$ that is independent in $\mathcal{M}'$, and satisfies the following property:

for any $j \notin J$,

$$\left|N_j^{G'}(\widetilde{S})\right| \geq (1-2\varepsilon)\cdot\min(\widetilde{t}_j - |N_j^G(v)|,\ t_j - |N_j^G(v)|) - \frac{\varepsilon(k-1)}{k^*}(t_j - |N_j^G(v)|) \quad (8.2)$$

Since $\widetilde{S}$ is independent in the matroid $\mathcal{M}'$, which is obtained by contracting $\mathcal{M}$ on v, we infer that $\widetilde{S}\cup\{v\}$ is an independent set in $\mathcal{M}$. It follows that the set $\widetilde{S}\cup\{v\}$ satisfies, for any $j \notin J$,

$$\begin{aligned}
\left|N_j^G(\widetilde{S}\cup\{v\})\right| &= \left|N^{G'}(\widetilde{S})\right| + \left|N_j^G(v)\right| \\
&\geq (1-2\varepsilon)\cdot\left(t_j - |N_j^G(v)|\right) - \frac{\varepsilon(k-1)}{k^*}t'_j + |N_j^G(v)| \\
&\geq (1-2\varepsilon)\cdot t_j - \frac{\varepsilon(k-1)}{k^*}t'_j \\
&\geq (1-2\varepsilon)\cdot\min(\widetilde{t}_j, t_j) - \frac{\varepsilon k}{k^*}t_j \qquad \text{(since } t'_j \leq t_j\text{)}
\end{aligned}$$

with (conditional) probability at least $\left(\frac{1}{Lk}\cdot\frac{\varepsilon}{2rd}\right)^{k-1}$. Thus, the unconditional probability that the set $\widetilde{S}\cup\{v\}$ satisfies the required property is at least

$$\frac{1}{2Lk}\cdot\left(\frac{1}{Lk}\cdot\frac{\varepsilon}{2rd}\right)^{k-1} \geq \left(\frac{1}{Lk}\cdot\frac{\varepsilon}{2rd}\right)^{k}.$$

Case 2. Suppose $v \notin S$. Now we consider two sub-cases.

Case 2.1: There exists a color j such that $\sum_{w'\in S}|N_j^G(v)\cap N_j^G(w')| \geq \varepsilon\cdot|N_j^G(v)|$.

We first claim that the probability that some vertex w from the set S is chosen to be u in Step 11 is at least $\frac{\varepsilon}{2rd}$. Then, conditioned on this event, we will use the induction hypothesis to show the required bound.

Fix a color $j \notin J$ satisfying the case assumption (if there are multiple such colors, pick one such color arbitrarily). Then, since color j satisfies the case assumption, it follows that,

$$\sum_{w'\in S} h_j(v) = \sum_{w'\in S}\frac{|N_j^G(w')\cap N_j^G(v)|}{|N_j^G(v)|} \geq \frac{1}{|N_j^G(v)|}\cdot\varepsilon\cdot|N_j^G(v)| = \varepsilon$$

Therefore,

$$\sum_{w'\in S} p(w') = \sum_{w'\in S}\frac{h(w')}{2rd} \geq \frac{1}{2rd}\cdot\sum_{w\in S} h_j(w') \geq \frac{1}{2rd}\cdot\varepsilon$$

Thus, the probability that some $w \in S$ will be chosen as the vertex u is at least $\frac{\varepsilon}{2rd}$. Now, we condition on this event. Then, by an argument similar to case 1, as $w \in S$, the set $\widetilde{S} \cup \{w\}$ is an independent set in $\mathcal{M}$ and covers at least

$$(1 - 2\varepsilon) \cdot \min(\widetilde{t}_j, t_j) - \frac{\varepsilon k}{k^*} t_j$$

for each $j \notin J$, with probability at least

$$\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd} \cdot \left(\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd}\right)^{k-1} = \left(\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd}\right)^{k}.$$

Case 2.2: For all colors j, $\sum_{w' \in S} |N_j(v) \cap N_j(w')| \leq \varepsilon \cdot |N_j(v)|$.

Hence,

$$\left| \left(\bigcup_{w' \in S} N_j(w') \right) \cap N_j(v) \right| \leq \sum_{w' \in S} |N_j(v) \cap N_j(w')| \leq \varepsilon \cdot |N_j(v)|. \tag{8.3}$$

Recall that the probability that the vertex u is equal to v is at least $1/2$, and we condition on this choice. Furthermore, recall that we have conditioned on the event that $A(\mathbf{v}) \cap S \neq \emptyset$, but due to case assumption $v \notin S$, there must exist a vertex $w \in A(\mathbf{v}) \cap S$.

In this case, we aim to show that v "approximately plays the role" of $w \in A(\mathbf{v}) \cap S$ in our solution. To this end, we consider different cases for the value of α_j in the vector $\mathbf{v}$. In each of the cases, we condition on the probability that the recursive call returns a set $\widetilde{S}$ with the desired properties. Using induction, this happens with probability at least $\left(\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd}\right)^{k-1}$. Thus, the unconditional probability is at least $\left(\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd}\right)^{k}$ as in Case 1. Moreover, Since $\widetilde{S}$ is independent in the matroid $\mathcal{M}'$, which is obtained by contracting $\mathcal{M}$ on v, we infer that $\widetilde{S} \cup \{v\}$ is an independent set in $\mathcal{M}$. Now we proceed to the analysis of each of the cases, conditioned on the good events.

Case A: $\alpha_j = 0$**.** Since $v \in A(j, 0)$, $|N_j^G(v)| \geq t_j$. Thus, $|N_j^G(\widetilde{S} \cup \{v\}| \geq t_j$, which is at least the claimed bound.

Case B: $\alpha_j = \lambda + 1$**.** Since $v, w \in A(j, \lambda + 1)$, $|N_j^G(v)| \leq \frac{\varepsilon t_j}{2k^*}$, and $|N_j^G(w)| \leq \frac{\varepsilon t_j}{2k^*}$.

Now we analyze $\left|N^{G'}(S \setminus \{w\})\right|$. By case assumption, it follows that,

$$\begin{aligned}\left|N_j^{G'}(S \setminus \{w\})\right| &= \widetilde{t}_j - |N_j^G(w)| - \left|\left(\bigcup_{w' \in S} N_j^G(w')\right) \cap N_j^G(v)\right| \\ &\geq \widetilde{t}_j - |N_j^G(w)| - |N_j^G(v)| \\ &\geq \widetilde{t}_j - \frac{2\varepsilon t_j}{2k^*} = \widetilde{t}_j - \frac{\varepsilon t_j}{k^*}.\end{aligned} \tag{8.4}$$

Then, by inductive hypothesis, the solution $\widetilde{S}$ satisfies the first inequality in the following.

$$\begin{aligned}\left|N_j^{G'}(\widetilde{S})\right| &\geq (1-2\varepsilon) \cdot \left(\widetilde{t}_j - \frac{\varepsilon t_j}{k^*}\right) - \frac{\varepsilon(k-1)t'_j}{k^*} \\ &\geq (1-2\varepsilon) \cdot \widetilde{t}_j - \frac{\varepsilon t_j}{k^*} - \frac{\varepsilon(k-1)t'_j}{k^*} \\ &\geq (1-2\varepsilon) \cdot \widetilde{t}_j - \frac{\varepsilon k}{k^*} t_j && \text{(Since } t'_j \leq t_j\text{)}\end{aligned}$$

which is at least the claimed bound.

Case C. $1 \leq \alpha_j \leq \lambda$.
Since $v, w \in A(j, \alpha_j)$, Observation 1 implies that

$$|N_j^G(w)| \leq (1+\varepsilon) \cdot |N_j^G(v)| \tag{8.5}$$

Analogous to (8.4), we have the following.

$$\begin{aligned}\left|N_j^{G'}(S \setminus \{w\})\right| &= \widetilde{t}_j - |N_j^G(w)| - \left|\left(\bigcup_{w' \in S} N_j^G(w')\right) \cap N_j^G(v)\right| \\ &\geq \widetilde{t}_j - |N_j^G(w)| - \varepsilon \cdot |N_j^G(v)| && \text{(From (8.3))} \\ &\geq \widetilde{t}_j - (1+2\varepsilon) \cdot |N_j^G(v)| && \text{(From (8.5))}\end{aligned}$$

Thus, by inductive hypothesis, it holds that,

$$\begin{aligned}|N_j^G(\widetilde{S} \cup \{v\})| &= |N_j^G(v)| + |N^{G'}(\widetilde{S})| \\ &\geq |N_j^G(v)| + (1-2\varepsilon) \cdot \left(\widetilde{t}_j - (1+2\varepsilon) \cdot |N_j^G(v)|\right) - \frac{\varepsilon(k-1)}{k^*} \cdot t_j \\ &\geq (1-2\varepsilon) \cdot \widetilde{t}_j + 4\varepsilon^2 \cdot |N_j^G(v)| - \frac{\varepsilon(k-1)}{k^*} \cdot t_j \\ &\geq (1-2\varepsilon) \cdot \widetilde{t}_j - \frac{\varepsilon(k-1)}{k^*} \cdot t_j && \text{(since } |N_j^G(v)| \geq 0\text{)}\end{aligned}$$

Running time: Note that the probability of a "good event" in the modified algorithm deteriorates by a maximum factor of $1/k$ at each branching step. This introduces an additional run time of k^k. And, the additional time taken to compute a representative family being polynomial-time for any bag $A(\mathbf{v})$ is absorbed into the polynomial factor of the algorithm's run time.

This completes the induction, and thus the proof of the lemma. □

We now state our main result.

Theorem 11. *There exists a randomized algorithm that runs in time* $\left(\frac{6kdr}{\varepsilon}\right)^k \cdot \left(\frac{18\log k}{\varepsilon^2}\right)^{kr} \cdot (n+m)^{\mathcal{O}(1)}$*, and given a yes-instance* $\mathcal{I} = (G, [r], f, \mathcal{M}, k, t)$ *of* (M, F)-Maximum Coverage*, where each element appears in at most* d *sets, with high probability, returns a subset* $\widetilde{S} \subseteq A$ *of size at most* k *with coverage vector* $(t''_1, t''_2, \ldots, t''_r)$*, such that* $t''_j \geq (1-\varepsilon)t_j$ *for all colors* j*; otherwise, if* $\mathcal{I}$ *is a no-instance, then the algorithm correctly concludes so.*

Proof. Let $S \subseteq A$ be a set of size k such that S is independent in $\mathcal{M}$ and $|N_j^G(S)| = \widetilde{t}_j \geq t_j$ for all colors j. Let $\tilde{S}$ denote the output of PMCCDS $(\mathcal{I})$. It is easy to see that the algorithm returns a solution in polynomial time. Next, Lemma 12 implies that with probability at least $q = \left(\frac{1}{Lk} \cdot \frac{\varepsilon}{2rd}\right)^k$, the set $\tilde{S}$ satisfies that, for each $j \in [r]$,

$$|N_j^G(\tilde{S})| = t''_j \geq (1-2\varepsilon) \cdot t_j - \frac{\varepsilon k}{k^*} t_j = (1-2\varepsilon)t_j - \varepsilon t_j = (1-3\varepsilon) \cdot t_j$$

and $\widetilde{S}$ is independent in $\mathcal{M}$. Here, we use the fact that since $\mathcal{I}$ is the original instance, we have $k^* = k$. Also note that, for any color $j \in [r], \widetilde{t}_j \geq t_j$.

We make $\mathcal{O}(q^{-1}\log n)$ independent calls to PMCCDS $(\mathcal{I})$, and if in any of the calls, we find a set $\tilde{S}$ with the claimed properties, then we return it. Otherwise, the algorithm concludes that $\mathcal{I}$ is a no instance. We get the claimed running time by rescaling ε to $\varepsilon/3$. □

Using the randomized reduction in Chapter 6 followed by the application of the EPAS in Theorem 11, we get the following result for (M, F)-Max k-weight SAT on formulas when either the size of the clauses is bounded by d or when every variable appears in at most d clauses.

Corollary 4. *Let* $\varepsilon > 0$*. Then,* (M, F)-Max k-weight SAT *admits a randomized EPAS with running time* $\left(\frac{6drk}{\varepsilon}\right)^k \cdot \left(\frac{18\log k}{\varepsilon^2}\right)^{kr} (n+m)^{\mathcal{O}(1)}$ *on formulas when either the*

size of the clauses is bounded by d or when every variable appears in at most d clauses. This result holds with constant probability.

Extension to Intersection of Multiple Linear Matroids. The above approach can be extended to the more general problem where the solution is required to be an independent set in each of the given matroids $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_q$ that are all defined over the common ground set A, *if all the given matroids are representable over a field* $\mathbf{F}$. The only change here is that, the representative family $R(A(\mathbf{v}))$ needs to be computed for the direct sum of the matroids $\mathcal{M}'_1 \oplus \mathcal{M}'_2 \oplus \cdots \oplus \mathcal{M}'_q$, where $\mathcal{M}'_i$ denotes the contracted version of the i-th matroid as defined above. Then, one can use the linear algebraic tools to compute a representative set of the direct sum matroid in a specific manner. For more details, we refer the reader to Marx ([36], Section 5.1). Due to this change, the bound on $R(A(\mathbf{v}))$ becomes qk (from k), and the time required to compute this set is at most $2^{\mathcal{O}(qk)} \cdot (m+n)^{\mathcal{O}(1)}$. This also gets reflected in the running time of the algorithm.

Chapter 9

A different parameter: number of satisfied constraints

We take a detour and study our problems with respect to a different parameter: the number of satisfied constraints. We design an FPT algorithm for CC-MaxSat on general instances in Chapter 9.1 and show that it admits a polynomial kernel on $K_{d,d}$-free instances in Chapter 9.2. To complete the picture, we also give a polynomial kernel for Maximum Coverage on $K_{d,d}$-free instances in Chapter 9.3.

9.1 FPT algorithm for CC-Max-SAT parameterized by number of satisfied constraints t

We present an FPT algorithm for CC-MaxSat problem parameterized by t, the number of satisfied clauses.

We use the algorithm for Partial Hitting Set given in Proposition 6 to get the following lemma.

Lemma 13. *There exists a randomized algorithm that solves the* CC-MaxSat *problem and outputs a satisying assignment in time* $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.

Proof. We apply the procedure in Algorithm 1, 2^t times and if in any of the iteration it returns an assignment γ, then we output γ, otherwise we return no instance.

Next we prove the correctness of the algorithm.

Algorithm 7 Procedure: FPT Algorithm for CC-MaxSat parameterized by t

1: Randomly assign values to all the variables in $\mathcal{V}_\phi$ from $\{0,1\}$. Let T and F be the set of variables assigned 1 and 0 respectively.
2: Delete all clauses C in $\mathcal{C}_\phi$ that contain a negative literal of a variable $v \in F$. Let t_1 be the number of deleted clauses and ϕ' be the resultant formula.
3: Construct a family $\mathcal{F}$ as follows.
4: **for** each clause $C \in \mathcal{C}_{\phi'}$, **do**
5: Add $\mathsf{var}(C) \setminus F$ to $\mathcal{F}$.
6: **end for**
7: Run the algorithm from Proposition 6 on the hitting set instance $(\mathcal{U}, \mathcal{F}, k, t-t_1)$ where $\mathcal{U} = \mathcal{V}_{\phi'} \setminus F$.
8: **if** the algorithm returns no instance, **then**
9: **return** no instance
10: **else** let S be the solution returned by the algorithm. We construct γ as follows.
11: **if** $v \in S$ **then**
12: let $\gamma(v) = 1$.
13: **else** $\gamma(v) = 0$.
14: **end if**
15: **return** γ.
16: **end if**

Let (ϕ, k, t) be a yes instance instance for the problem. Let $C_1, C_2, \cdots C_t$ be t clauses that are satisfied by a particular assignment, say α. For each clause C_i, let x_i be a variable that is "responsible" for satisfying it by a feasible assignment α, that is assignment where at most k variables are assigned 1. By "responsible" we mean that the clause C_i is satisfied even if we give any other assignment to all the variables except x_i. Thus, $x_1, x_2, \cdots x_t$ are the "responsible" variables for the clauses $C_1, C_2, \cdots C_t$ by the assignment α. Let α_t be the assignment α restricted to these t variables. Now consider any random assignment $\alpha' : \mathcal{V}_\phi \to \{0,1\}$ of $\mathcal{V}_\phi$. Then the assignments α and α' agree on the variables $x_1, x_2, \cdots x_t$ with probability $\frac{1}{2^t}$. Let α' be the assignment obtained in step 1 of our procedure. Therefore with probaility $\frac{1}{2^t}$ the assignment α' satisfies the clauses $C_1, C_2, \cdots C_t$.

Without loss of generality, let $C_1, C_2, \cdots C_{t_1}$ be the set of clauses deleted in Step 2 of our algorithm. Every clause C_i that is not deleted in Step 2 must contain one of the variables from $x_1, x_2, \cdots x_t$ which is assigned 1. There are at most k such variables and hence the corresponding Partial Hitting Set instance in Step 7 will return a solution. Since we repeat the algorithm 2^t times, we get a success probability of $1 - \frac{1}{e}$.

Running Time: Observe that the running time of Procedure FPT Algorithm for CC-MaxSat depends on the algorithm used in Step 7 which runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.

As we repeat the Procedure 2^t times, our algorithm runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$. □

We now derandomize the algorithm using universal sets[1]. We deterministically construct a family $\mathcal{F}$ of functions $f : [n] \to [2]$ instead of randomly assigning values such that it is assured that one of the assignments when restricted to the t variables $x_1, x_2, \ldots, x_t$ matches with the assignment α_t.

We construct an (n, t)-universal set $\mathcal{U}$ and then for every element of $\mathcal{U}$ we construct an equivalent assignment by assigning *true* to the variables represented by the elements in the subset and $false$ to the elements outside. By the definition of universal sets we will have an assignment which when restricted to the t responsible variables will be equal to α_t which corresponds to the good event in our random experiment. Thus, we have the following theorem.

Theorem 12. *There exists a deterministic algorithm that solves the* CC-MaxSat *problem in time* $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.

9.2 Polynomial Kernel for CC-MaxSat in $K_{d,d}$-free formulas parameterized by t

We next design a kernelization algorithm for CC-MaxSat where the input is a $K_{d,d}$-free formula, parameterized by the number of clauses to be satisfied by a solution – that is, the parameter is t.

9.2.1 Overview of the kernel

Consider an instance (Φ, k, t) of CC-MaxSat, where Φ is a $K_{d,d}$-free formula. Our kernelization algorithm works in three phases. In the first phase, we apply some simple sanity check reduction rules to eliminate trivial yes instance or no instance of CC-MaxSat. Reduction rules in this phase (1) upper bounds the frequency of variables in Φ by t, and (2) leads to an observation that any minimum weight assignment satisfying at least t clauses can satisfy at most $2t$ clauses. The above facts are useful to establish proofs in the next two phases.

Suppose (Φ, k, t) is a yes instance instance and let β be its minimum weight assignment and let $\mathcal{C}_\beta$ be the set of clauses satisfied by β. In the second phase, we bound

[1]Please check Definition 4 and Proposition 2 in Chapter 4 for details.

the size of clauses in Φ by a function of t and d, say $f(t,d)$. Here, we use $K_{d,d}$-free property crucially. By the definition of $K_{d,d}$-free formula, a set of d variables cannot appear simultaneously in a set of d clauses. We generalize this idea together with the frequency bound on variables (obtained in phase one) to bound the size of a set of variables that appear together in $p \in [d-1]$ clauses by identifying and deleting some "redundant" variables.

For a set $Y \subseteq \mathcal{V}_\Phi$, let $\mathsf{claInt}(Y)$ denote the set of all the clauses that contains all the variables in Y, that is the set $\{C \mid C \in \mathcal{C}_\Phi, Y \subseteq \mathsf{var}(C)\}$.

For an intuition, suppose we know that the size of every subset of $\mathcal{V}_\Phi$, that appear together in at least two clauses, is bounded by $f(t,d)$. In other words, there are at most $f(t,d)$ variables that can appear in two or more clauses together. Now consider a set $Y \subseteq \mathcal{V}_\Phi$ such that variables in Y appear together in at least one clause say C, that is $|\mathsf{claInt}(Y)| \geq 1$ and $C \in \mathsf{claInt}(Y)$. Now suppose that there is a clause $C^* \in \mathcal{C}_\beta \setminus \mathsf{claInt}(Y)$ such that $\mathsf{var}(C^*) \cap Y \neq \emptyset$. Then observe that for the variable set $\mathsf{var}(C^*) \cap Y$, C and C^* are common clauses. Now by considering the bound on sizes of sets of variables that have at least two common clauses (clauses in which they appear simultaneously), we obtain that $|\mathsf{var}(C^*) \cap Y|$ is also bounded by $f(t,d)$. We will now mark variables of all the clauses in $\mathcal{C}_\beta \setminus \mathsf{claInt}(Y)$ in Y and will conclude that if Y is sufficiently "large", then there exists a *redundant* variable in Y, that can be removed from Φ. Employing the above discussion, by repeatedly applying a careful deletion procedure, we manage to bound the size of variable sets that have at least one common clause. By repeating these arguments, we can show that to bound the size of variable sets with at least 2 common clauses, all we require is to bound size of variable sets with at least 3 common clauses, and so on. Thus, inductively, we bound size of variable sets with $d-1$ common clauses, by using the fact that the input formula is $K_{d,d}$-free.

Thus, the algorithm starts for $d-1$ and applies a reduction rule to bound size of sets of variables with $d-1$ common clauses. Once we apply reduction rule for $d-1$ exhaustively, we apply for $d-2$ and by inductive application we reach the one common clause case. To apply reduction rule for $p \in [d-1]$ we assume that reduction rules for $d-1$ common clauses case have already been applied exhaustively. The bound on size of the sets of variables with one common clause also gives a bound on the size of a clause by $f(t,d)$.

Finally in the third phase, the algorithm applies a reduction rule to remove all the variables which are not among first $g(t,d)$ high positive degree variables and not

among first $g(t,d)$ high negative degree variables when sorted in non decreasing ordering of their positive (negative) degrees, which together with the upper bound obtained on size frequency of variables and size of clauses obtained in phase one gives the desired kernel size.

The outline of the polynomial kernel for Decision Maximum Coverage is similar to that for Decision CC-MaxSat, therefore, we omit its overview.

9.2.2 Polynomial kernel for Decision CC-MaxSat on $K_{d,d}$-free formulae parameterized by t

We introduce our reduction rules below. Our algorithm applies each reduction rule exhaustively in the order in which they are stated. We begin by stating some simple sanity check reduction rules.

Reduction Rule 1. *If $k < 0$, or $k = 0$, $t \geq 1$ and $\beta_\emptyset$ satisfies less than t clauses in $\mathcal{C}_\Phi$, then return that (Φ, k, t) is a no instance instance of* CC-MaxSat.

The safeness of Reduction Rule 1 follows from the fact that the cardinality of number of variables assigned 1 cannot be negative and since $k = 0$, all the variables must be assigned 0 values and an all zero assignment must satisfy at least t clauses for (Φ, k, t) to be a yes instance instance of CC-MaxSat.

Reduction Rule 2. *If at least one of the following holds, then return that (Φ, k, t) is a yes instance instance of* CC-MaxSat.

1. *$\beta_\emptyset$ satisfies at least t clauses in $\mathcal{C}_\Phi$.*
2. *$k > 0$ and there exists a variable $v \in \mathcal{V}_\Phi$ such that $d^+_\Phi(v) \geq t$.*

The safeness of Reduction Rule 2 follows from the following facts: For the first condition, if an all zero assignment satisfies at least t clauses, then trivially (Φ, k, t) is a yes instance instance of CC-MaxSat. For the second condition, there exists an assignment which assigns v to true in case $d^+_\Phi(v) \geq t$, then the variable v alone can satisfy at least t clauses in $\mathcal{C}_\Phi$.

When none of the Reduction Rules 1 and 2 are applicable, we obtain the following observation.

Observation 4. *Consider a minimum weight assignment β of $\mathcal{V}_\Phi$ that satisfies at least t clauses in $\mathcal{C}_\Phi$. Then the number of clauses satisfied by β is at most $2t$.*

Proof. As Reduction Rule 2 is no longer applicable β is not an all zero assignment. Let u be a variable that has been assigned true value by β. Now consider another assignment β', where for every $u' \in \mathcal{V}_\Phi, u' \neq u$, $\beta'(u') = \beta(u')$, and $\beta'(u)$ is false. As Reduction Rule 2 is no longer applicable, β' is also not an all zero assignment. Also, β' satisfies strictly less than t clauses, as otherwise it contradicts that β is a minimum weight assignment. Moreover, as Reduction Rule 2 is no longer applicable, $d^+_\Phi(u) < t$ and $d^-_\Phi(u) < t$. Notice that the difference between clauses satisfied by β and β' are the following:

- clauses that contain u as a positive literal and the remaining literals in the clause evaluate to 0 - these type of clauses are satisfied under β only. Let the set of these clauses be $\mathcal{C}_{\text{upos}}$.
- clauses that contain u as a negative literal and the remaining literals in the clause evaluate to 0 - these type of clauses are satisfied under β' only. Let the set of these clauses be $\mathcal{C}_{\text{uneg}}$.

Then, we can easily observe the following.

$$t > |\mathcal{C}_{\beta'}| \geq |\mathcal{C}_\beta| - |\mathcal{C}_{\text{upos}}| + |\mathcal{C}_{\text{uneg}}|.$$

Above observations imply that β satisfies at most $t + |\mathcal{C}_{\text{upos}}| - |\mathcal{C}_{\text{uneg}}| \leq t + |\mathcal{C}_{\text{upos}}| \leq 2t$ clauses. □

Lemma 14. *Consider a set $Y \subseteq \mathcal{V}_\Phi$. Let $|\mathsf{cla Int}(Y)| = \ell$. If $|Y| \geq 2^\ell \cdot \tau + 1$, for some positive integer τ, then in polynomial time we can find sets $\widehat{Y}_{\mathsf{pos}}, \widehat{Y}_{\mathsf{neg}} \subseteq Y$ and a variable $\widehat{v} \in Y \setminus (\widehat{Y}_{\mathsf{pos}} \cup \widehat{Y}_{\mathsf{neg}})$ such that the following holds:*

1. $|\widehat{Y}_{\mathsf{pos}}| = |\widehat{Y}_{\mathsf{neg}}| = \frac{\tau}{2}$.
2. *Let $\widehat{Y} = \widehat{Y}_{\mathsf{pos}} \cup \widehat{Y}_{\mathsf{neg}} \cup \{\widehat{v}\}$. For every pair of variables $u, u' \in \widehat{Y}$ and every clause $C \in \mathsf{cla Int}(Y)$, u appears in C positively (negatively) if and only if u' appears in C positively (negatively).*
3. *For every variable $u \in \widehat{Y}_{\mathsf{pos}}$, $d^+_\Phi(\widehat{v}) \leq d^+_\Phi(u)$.*
4. *For every variable $u \in \widehat{Y}_{\mathsf{neg}}$, $d^-_\Phi(\widehat{v}) \leq d^-_\Phi(u)$.*

Proof. Let $\mathsf{cla Int}(Y) = \{C_1, \cdots, C_\ell\}$. For each $u \in Y$, we define a string Γ_u on $\{0, 1\}$ of length ℓ by setting i-th bit of Γ_u as 1 (0) if u appears as positively (negatively)

in C_i. By simple combinatorial arguments, we have that the number of different Γ strings that we can obtain are at most 2^ℓ. Since $|Y| \geq 2^\ell \cdot \tau + 1$, by pigeonhole principle there exists a set $Y' \subseteq Y$ of size at least $\tau + 1$ such that for every pair u, u' of variables in Y', $\Gamma_u = \Gamma_{u'}$. Also for every clause $C \in \mathsf{cla Int}(Y)$, u appears in C positively (negatively) if and only if u' appears in C positively (negatively). Now we obtain $\widehat{Y}_{\mathsf{pos}}$ and $\widehat{Y}_{\mathsf{neg}}$ from Y'.

We let $\widehat{Y}_{\mathsf{pos}}$ be a subset of Y' of size $\frac{\tau}{2}$ such that for every $u \in \widehat{Y}_{\mathsf{pos}}$ and every $u' \in Y' \setminus \widehat{Y}_{\mathsf{pos}}$, $d^+_\Phi(u') \leq d^+_\Phi(u)$. We let $\widehat{Y}_{\mathsf{neg}}$ be a subset of Y' of size $\frac{\tau}{2}$ such that for every $u \in \widehat{Y}_{\mathsf{neg}}$ and every $u' \in Y' \setminus \widehat{Y}_{\mathsf{neg}}$, $d^-_\Phi(u') \leq d^-_\Phi(u)$, that is, $\widehat{Y}_{\mathsf{pos}}$ is the set of first $\frac{\tau}{2}$ variables in Y' when sorted by their positive degrees. Similarly $\widehat{Y}_{\mathsf{neg}}$ is the set of first $\frac{\tau}{2}$ variables in Y' when sorted by their negative degrees.

Observe that since $|\widehat{Y}_{\mathsf{pos}}| + |\widehat{Y}_{\mathsf{neg}}| < |Y'|$, therefore $Y' \setminus (\widehat{Y}_{\mathsf{pos}} \cup \widehat{Y}_{\mathsf{neg}}) \neq \emptyset$. We let $\widehat{v}$ be an arbitrary variable in $Y' \setminus (\widehat{Y}_{\mathsf{pos}} \cup \widehat{Y}_{\mathsf{neg}})$. By the above description, clearly $\widehat{v}, \widehat{Y}_{\mathsf{pos}}$, and $\widehat{Y}_{\mathsf{neg}}$ satisfies the properties stated in lemma and can be computed in polynomial time. □

Next, we will describe reduction rules that help us bound the size of clauses in $\mathcal{C}_\Phi$. For each $p \in [d-1]$, we introduce Reduction Rule 3.p. We apply Reduction Rule 3.p in the increasing order of p. That is, first apply Reduction Rule 3.1 exhaustively, and for each $p \in [d-1] \setminus \{1\}$, apply Reduction Rule 3.p only if Reduction Rule 3.$(p-1)$ has been applied exhaustively. We apply our reduction rule on a "large" subset of $\mathcal{V}_\Phi$. To quantify "large" we introduce the following definition.

Definition 12. *For each $p \in [d-1]$, we define an integer z_p as follows:*

- *If $p = 1$, then $z_1 = 2^{d+1} \cdot (t(d-1)+1)$, and*
- *$z_p = 2^{d-p+2} \cdot (t \cdot z_{p-1} + 1)$, otherwise.*

The following observation will be helpful to bound the size of our kernel.

Observation 5. $z_{d-1} \leq 4^{d^2}(d \cdot t^d + 1)$.

Reduction Rule 3. *For each $p \in [d-1]$ we introduce* Reduction Rule 3.p *as follows. If there exists $Y \subseteq \mathcal{V}_\Phi$ such that $|\mathsf{cla Int}(Y)| = d - p$ and $|Y| = z_p + 1$. Use Lemma 14 to find sets $\widehat{Y}_{\mathsf{pos}}, \widehat{Y}_{\mathsf{neg}} \subseteq Y$ and a variable $\widehat{v} \in Y \setminus (\widehat{Y}_{\mathsf{pos}} \cup \widehat{Y}_{\mathsf{neg}})$ which satisfies properties stated in Lemma 14. Remove $\widehat{v}$ from $\mathcal{V}_\Phi$ and return the instance (Φ', k, t). Here, Φ' is the formula with variable set $\mathcal{V}_\Phi \setminus \{\widehat{v}\}$ and clause set $\bigcup_{C \in \mathcal{C}_\Phi} C \setminus \{\widehat{v}\}$.*

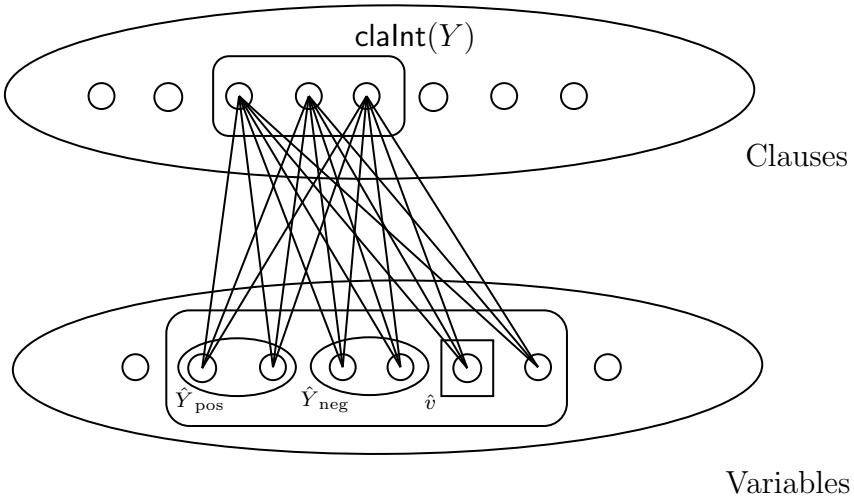

Figure 9.1: A visual representation of Reduction Rule 3

Lemma 15. *Reduction Rule 3 is safe.*

Proof. To show that the lemma holds, we will show that (Φ, k, t) is a yes instance instance of CC-MaxSat if and only if (Φ', k, t) is a yes instance instance of CC-MaxSat, for each $p \in [d-1]$. We prove the lemma by induction on p.

Base Case: $p = 1$. We have $z_1 = 2^{d+1} \cdot (t(d-1)+1)$. By Lemma 14 we have $|\widehat{Y}_{\mathsf{pos}}| = |\widehat{Y}_{\mathsf{neg}}| = 2t(d-1)+2$. In the forward direction suppose that (Φ, k, t) is a yes instance instance of CC-MaxSat and let β be its minimum weight assignment. Let $\mathcal{C}_\beta \subseteq \mathcal{C}_\Phi$ be the set of clauses satisfied by β and $\widetilde{\mathcal{C}_\beta} = \mathcal{C}_\beta \setminus \mathsf{claInt}(Y)$ be the set of clauses satisfied by β but not in $\mathsf{claInt}(Y)$. Observe the following:

1. By Observation 4, the number of clauses satisfied by β is at most $2t$, that is $|\mathcal{C}_\beta| \leq 2t$.
2. Suppose that $C \in \widetilde{\mathcal{C}_\beta}$. Then, since Φ is a $K_{d,d}$-free formula and $|\mathsf{claInt}(Y) \cup \{C\}| = d$, we have $|\mathsf{var}(C) \cap Y| \leq d-1$. As otherwise the set of variables $\mathsf{var}(C) \cap Y$ and set of clauses $\mathsf{claInt}(Y) \cup \{C\}$ will contradict $K_{d,d}$-free property.

By (1) and (2) the set of variables in Y, that appear in clauses in the set $\widetilde{\mathcal{C}_\beta}$ is bounded by $2t(d-1)$. Hence the set of variables in $\widehat{Y}_{\mathsf{pos}}$ and in $\widehat{Y}_{\mathsf{neg}}$, that appear in clauses in the set $\widetilde{\mathcal{C}_\beta}$ is bounded by $2t(d-1)$. That is $|\bigcup_{C \in \widetilde{\mathcal{C}_\beta}} \mathsf{var}(C) \cap \widehat{Y}_{\mathsf{pos}}| \leq 2t(d-1)$ and $|\bigcup_{C \in \widetilde{\mathcal{C}_\beta}} \mathsf{var}(C) \cap \widehat{Y}_{\mathsf{neg}}| \leq 2t(d-1)$.

Recall that $|\widehat{Y}_{\mathsf{pos}}| = |\widehat{Y}_{\mathsf{neg}}| = 2t(d-1)+2$. Therefore, there exists two variables say $w_1, w_2 \in \widehat{Y}_{\mathsf{pos}}$ such that w_1, w_2 do not appear in any clause in $\widetilde{\mathcal{C}_\beta}$. Similarly, there

exists two variables say $u_1, u_2 \in \widehat{Y}_{\mathsf{neg}}$ such that u_1, u_2 do not appear in any clause in $\widetilde{\mathcal{C}_\beta}$. That is every clause that is satisfied by β and contains any of w_1, w_2, u_1, u_2 is contained in $\mathsf{clalnt}(Y)$. This also implies that every clause that is satisfied by any of the variables w_1, w_2, u_1, u_2 in assignment β, is contained in $\mathsf{clalnt}(Y)$. Now consider the following cases:

Case 1: $\beta(\widehat{v}) = 1$. First we claim that w_1 and w_2 are both set to false by β. Suppose for a contradiction that one of them say w_1 is set to true, that is $\beta(w_1) = 1$. By the properties of $\widehat{Y}_{\mathsf{pos}}$, and $\widehat{v}$ (See Lemma 14), in every clause $C \in \mathsf{clalnt}(Y)$, either both $\widehat{v}$ and w_1 appear positively in C or both $\widehat{v}, w_1$ appear negatively in C. Further, since w_1 can only satisfy clauses in $\mathsf{clalnt}(Y)$ by assignment β, we have that w_1 satisfies same set of clauses as $\widehat{v}$ in $\mathsf{clalnt}(Y)$. That is $N_\Phi(w_1) \cap \mathcal{C}_\beta = N_\Phi(\widehat{v}) \cap \mathcal{C}_\beta$. In this case, we can obtain another assignment of smaller weight by setting w_1 to false, which contradicts that β is a minimum weight assignment. By similar arguments we can show that w_2 is set to false by β.

Now we will construct an assignment β' for variables $\mathcal{V}_{\Phi'} = \mathcal{V}_\Phi \setminus \{\widehat{v}\}$, by setting the value of w_1 to true and we will show that β' satisfies as many clauses in Φ' as satisfied by β in Φ. We define β' formally as follows: $\beta'(w_1) = 1$, and for every $u \in \mathcal{V}_\Phi \setminus \{\widehat{v}, w_1\}$, $\beta'(u) = \beta(u)$. Notice that $\mathcal{C}_\beta$ comprises of the following set of clauses satisfied by β (i) clauses that do not contain variables $\widehat{v}, w_1$, (ii) clauses that are in $\mathsf{clalnt}(Y)$ and that contains variable $\widehat{v}$ positively, (iii) clauses that are in $\mathsf{clalnt}(Y)$ and that contains variable w_1 negatively, and lastly (iv) clauses that are not in $\mathsf{clalnt}(Y)$ and that contains variable $\widehat{v}$ positively.

Clearly $\mathcal{C}_{\Phi'}$ contains every clause in the set (i) and they are also satisfied by β'. For every clause $C \in \mathcal{C}_\Phi$ that contain $\widehat{v}$, we have a clause $C \setminus \{\widehat{v}\}$ in $\mathcal{C}_{\Phi'}$. If C is a clause in the set (ii), then C contains $\widehat{v}$ positively and thus also contains w_1 positively. Therefore, $C \setminus \{\widehat{v}\}$ is satisfied by β'. Hence for every clause in the set (ii), we have a clause in $\mathcal{C}_{\Phi'}$ satisfied by w_1 in β'. Next, consider a clause C in the set (iii), then C contains w_1 negatively and thus also contains w_2 negatively. As argued before $\beta(w_2) = \beta'(w_2) = 0$. Therefore, $C \setminus \{\widehat{v}\}$ is satisfied by w_2 in β'. Hence for every clause in set (iii), we have a clause in $\mathcal{C}_{\Phi'}$ satisfied by β'.

Now we are only remaining to show that β' compensate for the clauses in the set (iv) for Φ'. For that purpose recall that we have $d^+_\Phi(\widehat{v}) \leq d^+_\Phi(w_1)$, by the properties of $\widehat{Y}_{\mathsf{pos}}$, and $\widehat{v}$ (See Lemma 14). Therefore $|N^+_\Phi(w_1)| \geq |N^+_\Phi(\widehat{v})|$, and hence $|N^+_{\Phi'}(w_1)| \geq |N^+_\Phi(\widehat{v})|$. Also $(N^+_\Phi(w_1) \cap \mathcal{C}_\beta) \setminus \mathsf{clalnt}(Y) = \emptyset$. That is, the clauses which contains w_1 positively and are not in $\mathsf{clalnt}(Y)$ were not satisfied by β. As β' sets w_1 to true,

now clauses in $N^+_{\Phi'}(w_1)$ are satisfied by β'. We obtain the following:

$$|(N^+_{\Phi}(w_1) \cap \mathcal{C}_{\beta})| - |N^+_{\Phi}(w_1) \cap \mathsf{claInt}(Y)| \geq |(N^+_{\Phi}(\widehat{v}) \cap \mathcal{C}_{\beta})| - |N^+_{\Phi}(\widehat{v}) \cap \mathsf{claInt}(Y)|.$$

$$|(N^+_{\Phi'}(w_1) \cap \mathcal{C}_{\beta'})| - |N^+_{\Phi}(w_1) \cap \mathsf{claInt}(Y)| \geq |(N^+_{\Phi}(\widehat{v}) \cap \mathcal{C}_{\beta})| - |N^+_{\Phi}(\widehat{v}) \cap \mathsf{claInt}(Y)|.$$

All the above discussion concludes that the number of clauses satisfied by β' in Φ' are at least the number of clauses satisfied by β in Φ. Hence, (Φ', k, t) is a yes instance instance of CC-MaxSat.

Case 2: $\beta(\widehat{v}) = 0$. We can show that at least one of u_1, u_2 is set to false by β, by using analogous arguments that were used in Case 1 to show that when $\widehat{v}$ is set to true then both w_1 and w_2 are set to false. Without loss of generality suppose that u_1 is set to false by β. We will show that assignment β restricted to $\mathcal{V}_{\Phi'} = \mathcal{V}_{\Phi} \setminus \{\widehat{v}\}$ satisfies as many clauses in Φ' as satisfied by β in Φ. We define β' as β restricted to $\mathcal{V}_{\Phi'}$ formally as follows: for every $u \in \mathcal{V}_{\Phi} \setminus \{\widehat{v}\}$, $\beta'(u) = \beta(u)$. Notice that $\mathcal{C}_{\beta}$ comprises of the following set of clauses satisfied by β (i) clauses that do contain variable $\widehat{v}$, (ii) clauses that are in $\mathsf{claInt}(Y)$ and that contains variable $\widehat{v}$ negatively, and lastly (iii) clauses that are not in $\mathsf{claInt}(Y)$ and that contains variable $\widehat{v}$ negatively.

Clearly $\mathcal{C}_{\Phi'}$ contains every clause in the set (i) and they are also satisfied by β'. For every clause $C \in \mathcal{C}_{\Phi}$ that contain $\widehat{v}$, we have a clause $C \setminus \{\widehat{v}\}$ in $\mathcal{C}_{\Phi'}$. If C is a clause in the set (ii), then C contains $\widehat{v}$ negatively and thus also contains u_1 negatively. Therefore, $C \setminus \{\widehat{v}\}$ is satisfied by β'. Hence for every clause in the set (ii), we have a clause in $\mathcal{C}_{\Phi'}$ satisfied by u_1 in β'. Now we are only remaining to consider the clauses in the set (iii). That is the set of clauses that are not in $\mathsf{claInt}(Y)$ and contains $\widehat{v}$ negatively. We claim that there is no clause in set (iii), that is $N^-_{\Phi}(\widehat{v}) \setminus \mathsf{claInt}(Y) = \emptyset$.

To prove the claim first recall that we have $d^-_{\Phi}(\widehat{v}) \leq d^-_{\Phi}(u_1)$, by the properties of $\widehat{Y}_{\mathsf{neg}}$, and $\widehat{v}$ (See Lemma 14). Therefore $|N^-_{\Phi}(u_1)| \geq |N^-_{\Phi}(v)|$. Also $N^-_{\Phi}(u_1) \cap \mathsf{claInt}(Y) = N^-_{\Phi}(\widehat{v}) \cap \mathsf{claInt}(Y)$. Further, $(N^-_{\Phi}(u_1) \cap \mathcal{C}_{\beta}) \setminus \mathsf{claInt}(Y) = \emptyset$. Therefore, $N^-_{\Phi}(u_1) \setminus \mathsf{claInt}(Y) = \emptyset$, as u_1 is set to false by β. If $(N^-_{\Phi}(\widehat{v}) \cap \mathcal{C}_{\beta}) \setminus \mathsf{claInt}(Y) \neq \emptyset$, then $N^-_{\Phi}(\widehat{v}) \setminus \mathsf{claInt}(Y) \neq \emptyset$. Therefore $|N^-_{\Phi}(u_1) \setminus \mathsf{claInt}(Y)| < |N^-_{\Phi}(\widehat{v}) \setminus \mathsf{claInt}(Y)|$. Hence, the following holds:

$$d^-_{\Phi}(u_1) = |(N^-_{\Phi}(u_1) \cap \mathsf{claInt}(Y))| + |(N^-_{\Phi}(u_1) \cap \mathsf{claInt}(Y))|,$$

$$d^-_{\Phi}(u_1) = |(N^-_{\Phi}(u_1) \cap \mathsf{claInt}(Y))| + |(N^-_{\Phi}(\widehat{v}) \cap \mathsf{claInt}(Y))|,$$

$$d^-_{\Phi}(u_1) < |(N^-_{\Phi}(\widehat{v}) \cap \mathsf{claInt}(Y))| + |(N^-_{\Phi}(\widehat{v}) \cap \mathsf{claInt}(Y))|.$$

Thus $d^-_\Phi(u_1) < d^-_\Phi(\widehat{v})$, a contradiction. All the above discussion concludes that the number of clauses satisfied by β' in Φ' are at least the number of clauses satisfied by β in Φ. Hence, (Φ', k, t) is a yes instance instance of CC-MaxSat.

It is easy to see that in the backward direction if (Φ', k, t) is a yes instance instance of CC-MaxSat then (Φ, k, t) is a yes instance instance of CC-MaxSat. As any assignment β' of $\mathcal{V}_{\Phi'}$ can be extended to an assignment β of Φ by setting $\widehat{v}$ to false and assigning every variable $u \neq \widehat{v}$ as $\beta(u)$. By the definition of Φ', β satisfies as many clauses as β' and weight of β is equal to weight of β'.

Induction Hypothesis: Assume that Reduction Rule 3.p is safe for all $p < q$, $q \in [d-2]$.

Inductive Case: $p = q$. We have $z_p = 2^{d-p+2} \cdot (t \cdot z_{p-1} + 1) + 1$. By Lemma 14 we have $|\widehat{Y}_{\mathsf{pos}}| = |\widehat{Y}_{\mathsf{neg}}| = 2t \cdot z_{p-1} + 2$. In the forward direction suppose that (Φ, k, t) is a yes instance instance of CC-MaxSat and let β be its minimum weight assignment. Let $\mathcal{C}_\beta \subseteq \mathcal{C}_\Phi$ be the set of clauses satisfied by β and $\widetilde{\mathcal{C}}_\beta = \mathcal{C}_\beta \setminus \mathsf{claInt}(Y)$ be the set of clauses satisfied by β but not in $\mathsf{claInt}(Y)$. Observe the following:

1. By Observation 4, the number of clauses satisfied by β is at most $2t$, that is $|\mathcal{C}_\beta| \leq 2t$.
2. Suppose that $C \in \widetilde{\mathcal{C}}_\beta$. Then, since Reduction Rule 3.$(p-1)$ is not applicable and $|\mathsf{claInt}(Y) \cup \{C\}| = d - p + 1 = d - (p-1)$, we have $|\mathsf{var}(C) \cap Y| \leq z_{p-1}$. As otherwise the set of variables $\mathsf{var}(C) \cap Y$ and set of clause $\mathsf{claInt}(Y) \cup \{C\}$ will contradict that Reduction Rule 3.$(p-1)$ is not applicable.

By (1) and (2) the number of variables in Y, that appear in clauses in the set $\widetilde{\mathcal{C}}_\beta$ is bounded by $2t \cdot z_{p-1}$. Hence the number of variables in $\widehat{Y}_{\mathsf{pos}}$ and in $\widehat{Y}_{\mathsf{neg}}$, that appear in clauses in the set $\widetilde{\mathcal{C}}_\beta$ is bounded by $2t \cdot z_{p-1}$. That is $|\bigcup_{C \in \widetilde{\mathcal{C}}_\beta} \mathsf{var}(C) \cap \widehat{Y}_{\mathsf{pos}}| \leq 2t \cdot z_{p-1}$ and $|\bigcup_{C \in \widetilde{\mathcal{C}}_\beta} \mathsf{var}(C) \cap \widehat{Y}_{\mathsf{neg}}| \leq 2t \cdot z_{p-1}$.

Recall that $|\widehat{Y}_{\mathsf{pos}}| = |\widehat{Y}_{\mathsf{neg}}| = 2t \cdot z_{p-1} + 2$. Therefore, there exists two variables say $w_1, w_2 \in \widehat{Y}_{\mathsf{pos}}$ such that w_1, w_2 do not appear in any clause in $\widetilde{\mathcal{C}}_\beta$. Similarly, there exists two variables say $u_1, u_2 \in \widehat{Y}_{\mathsf{neg}}$ such that u_1, u_2 do not appear in any clause in $\widetilde{\mathcal{C}}_\beta$. Now by analogous arguments as in Case 1 and Case 2 of the base case, it follows that (Φ', k, t) is a yes instance instance of CC-MaxSat. Backward direction also follows similar to the base case. This completes the proof. □

When Reduction Rule 3 is no longer applicable, we have that every set $Y \subseteq \mathcal{V}_\Phi$ such

that $|\mathsf{claInt}(Y)| \geq d - (d-1) = 1$, satisfies $|Y| \leq z_{d-1}$. In other words for every clause $C \in \mathcal{C}_\Phi$, $|\mathcal{V}_\Phi| \leq z_{d-1}$. We record this observation in the following.

Observation 6. *When Reduction Rules 1-3 are not applicable, then for every clause $C \in \mathcal{C}_\Phi$, $|\mathsf{var}(C)| \leq z_{d-1}$.*

For stating our next reduction rule, we define two sets $\widehat{V}_{\mathsf{pos}}, \widehat{V}_{\mathsf{neg}} \subseteq \mathcal{V}_\Phi$ of size $\min\{n, z_{d-1}+1\}$ with the following properties: (1) For every variable $u \in \widehat{V}_{\mathsf{pos}}$ and every variable $u' \in \mathcal{V}_\Phi \setminus \widehat{V}_{\mathsf{pos}}$, $d^+_\Phi(u) \geq d^+_\Phi(u')$. (2) For every variable $u \in \widehat{V}_{\mathsf{neg}}$ and every variable $u' \in \mathcal{V}_\Phi \setminus \widehat{V}_{\mathsf{neg}}$, $d^-_\Phi(u) \geq d^-_\Phi(u')$. Clearly, if $\mathcal{V}_\Phi \geq 2t \cdot z_{d-1} + 3$, then by pigeonhole principle $\mathcal{V}_\Phi \setminus (\widehat{V}_{\mathsf{pos}} \cup \widehat{V}_{\mathsf{neg}}) \neq \emptyset$.

Reduction Rule 4. *If $|\mathcal{V}_\Phi| \geq 2t \cdot z_{d-1} + 3$, then let $u \in \mathcal{V}_\Phi \setminus (\widehat{V}_{\mathsf{pos}} \cup \widehat{V}_{\mathsf{neg}})$. Remove u from $\mathcal{V}_\Phi$ and return the instance (Φ', k, t). Here Φ' is the formula with variable set $\mathcal{V}_\Phi \setminus \{u\}$ and clause set $\bigcup_{C \in \mathcal{C}_\Phi} C \setminus \{u\}$.*

Lemma 16. *Reduction Rule 4 is safe.*

Proof. In the forward direction suppose that (Φ, k, t) is a yes instance instance of CC-MaxSat. Let β be its minimum weight assignment and let X be the set of clauses satisfied by β. Since Reduction Rule 2 is not applicable β is not an all zero assignment.

(1) By Observation 4, $|X| \leq 2t$.

(2)By Observation 6, for every $C \in \mathcal{C}_\Phi$, $|\mathsf{var}(C)| \leq z_{d-1}$.

Let $Y = \bigcup_{C \in X} \mathsf{var}(C)$, then by (1) and (2), $|Y| \leq 2t \cdot z_{d-1}$. Therefore, there exists a variable say $w_1 \in \widehat{V}_{\mathsf{pos}}$ and there exists a variable say $w_2 \in \widehat{V}_{\mathsf{neg}}$ such that $w_1, w_2 \notin Y$ and hence, $N_\Phi(w_1) \cap X = N_\Phi(w_2) \cap X = \emptyset$. Since β is a minimum weight assignment, $\beta(w_1) = \beta(w_2) = 0$.

Case 1: $\beta(u) = 1$. We obtain another assignment by setting the value of w_1 to true. That is we construct a new assignment β' of $\mathcal{V}_{\Phi'} = \mathcal{V}_\Phi \setminus \{u\}$ as follows: $\beta'(w_1) = 1$, and for every $v \in \mathcal{V}_\Phi \setminus \{u, w_1\}$, $\beta'(v) = \beta(v)$. We have $d^+_\Phi(u) \leq d^+_\Phi(w_1)$ (by the definition of $\widehat{V}_{\mathsf{pos}}$). Let $X' = (X \setminus N^+_\Phi(u)) \cup N^+_\Phi(w_1)$ and $X'' = \bigcup_{C \in X'} C \setminus \{u\}$. Then observe that X'' is the set of clauses satisfied by β'. Also $|X''| = |X'| \geq |X| \geq t$. This implies that β' is a solution to (Φ', k, t) of CC-MaxSat.

Case 2: $\beta(u) = 0$. We have $d^-_\Phi(u) \leq d^-_\Phi(w_2)$ (by the definition of $\widehat{V}_{\mathsf{neg}}$) and $N^-_\Phi(w_2) \cap X = \emptyset$. As both u, w_2 are set to false by β, $N^-_\Phi(w_2) \setminus X = N^-_\Phi(u) \setminus X = \emptyset$. Therefore $N^-_\Phi(u) \cap X = \emptyset$, as otherwise $|N^-_\Phi(u) \cap X| > |N^-_\Phi(w_2) \cap X|$ which

contradicts $d^-_\Phi(u) \leq d^-_\Phi(w_2)$. Then observe that X'' is the set of clauses satisfied by β restricted to $\mathcal{V}_\Phi \setminus \{u\} = \mathcal{V}_{\Phi'}$. Also $|X''| = |X'| = |X| \geq t$. This implies that assignment β restricted to $\mathcal{V}_{\Phi'}$ is a solution to (Φ', k, t) of CC-MaxSat.

It is easy to see that in the backward direction if (Φ', k, t) is a yes instance instance of CC-MaxSat then (Φ, k, t) is a yes instance instance of CC-MaxSat. As any assignment β' of $\mathcal{V}_\Phi$ can be extended to an assignment β of Φ by setting u to false and assigning every variable $v \neq u$ as $\beta(v)$. Observe that β satisfies as many clauses as β' and weight of β is equal to weight of β'. This completes the proof. □

Observe that Reduction Rules 1, 2 and 4 can be applied in polynomial time. Reduction Rule 3 can be applied in $m^{\mathcal{O}(d)}n^{\mathcal{O}(1)}$ time in the following manner: to apply Reduction Rule 3.p, for some $p \in [d-1]$, try all subsets of clauses of size $d-p$. For each such subset, check if the set of variables in common in these clauses is at least $z_p + 1$ (this check can be done in time polynomial in the size of the clauses). If so, apply Reduction Rule 3.p on this set.

Each of our reduction rules is applicable only polynomially many times. Hence, the kernelization algorithm runs in polynomial time. Each of our reduction rules is safe. When Reduction Rules 1-4 are no longer applicable, the size of the set $\mathcal{V}_\Phi$ is bounded by $2t(z_{d-1}) + 2$, and number of clauses a variable appears in is bounded by $2t$, and there are no variables which do not appear in any clause. Therefore by using Observation 4, the number of variables and clauses in Φ is bounded by $\mathcal{O}(d4^{d^2+1}t^{d+1})$.

Theorem 13. CC-Max-SAT *in $K_{d,d}$-free formulae admits a kernel of size $\mathcal{O}(d4^{d^2}t^{d+1})$.*

9.3 Polynomial kernel for Dec MaxRBDS on $K_{d,d}$-free graphs parameterized by t

We design a kernelization algorithm for Dec MaxRBDS on $K_{d,d}$-free graphs, parameterized by the number of vertices to be dominated by a solution – i.e., the parameter is t. This is also an important component used in improving the running time of the EPAS of MaxRBDS in Chapter 5.2.

We give an outline of the kernel below. Consider an instance $(G = (A, B, E), k, t)$ of Dec MaxRBDS, where G is a $K_{d,d}$-free bipartite graph with bipartition (A, B), and $k, t \in \mathbb{N}$. Our kernelization algorithm works in three phases. In the first

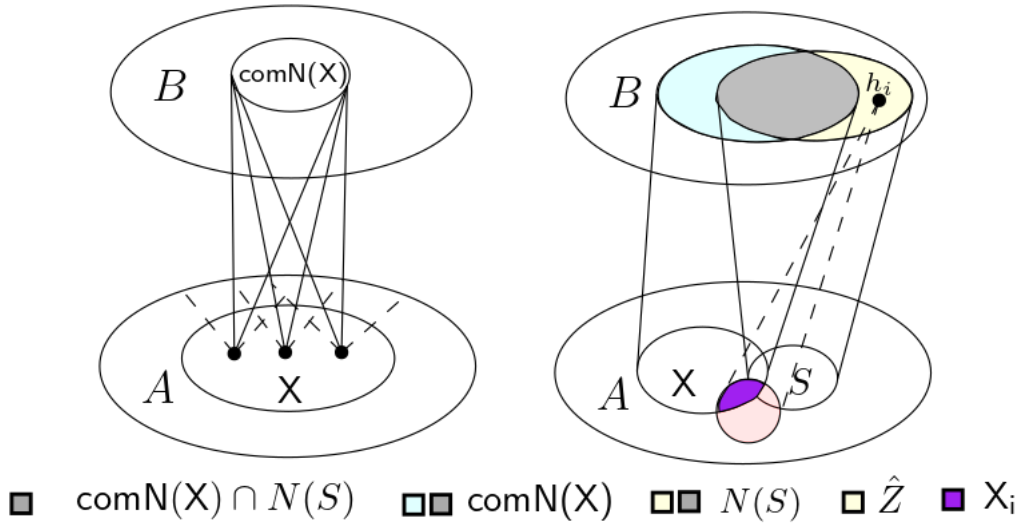

Figure 9.2: Notations for Kernel. The union of colors in boxes denotes the corresponding sets

phase, we apply some simple sanity check reduction rules to eliminate trivial yes instance/no instance instances of Dec MaxRBDS and a reduction rule to remove all the isolated vertices in G. Reduction rules in this phase (1) upper bound the degree of vertices in A by t, (2) lower bound degree of vertices in G by 1, and (3) lead to an observation that any smallest solution S, can dominate at most $2t$ vertices. The above facts are useful to establish proofs in the next two phases.

Suppose (G, k, t) is a yes instance instance and let S be a smallest solution.

In the second phase, we bound the degree of vertices in B by a function of t and d, say $f(t, d)$. Here, we use the $K_{d,d}$-free property crucially. By the definition of $K_{d,d}$-free graph, a set of size d in A cannot have d common neighbors in B. In other words, a set of size d in B cannot have d common neighbors in A. We generalize this idea together with the degree bound on A (obtained in phase one) to bound the size of a set in A that have $p \in [d-1]$ *common neighbors* by identifying and deleting some "irrelevant" vertices in A. For a set $X \subseteq A$, let $\mathsf{comN}(X)$ denote $\bigcap_{u \in X} N(u)$. Let $\mathsf{comN}(X, S) = \mathsf{comN}(X) \cap N(S)$, and $\widehat{Z} = N(S) \setminus \mathsf{comN}(X)$. Let $N(S) = \{h_1, \ldots, h_t\}$. For each vertex $h_i \in N(S)$, let $X_i = X \cap N(h_i)$. Refer to Figure 9.2 for a visual representation.

For an intuition, suppose we have already managed to bound the size of every subset of A, that has at least two common neighbors, by $f(t, d)$. Now consider a set $X \subseteq A$ such that vertices in X has at least one common neighbor say u, i.e., $u \in \mathsf{comN}(X)$.

Now, suppose that a vertex $h_i \in (N(S) \setminus \{u\})$ has neighbors in X, that is $X_i =$

$N(h_i) \cap X \neq \emptyset$. Then observe that vertices in X_i have at least two common neighbors including u and h_i, that is $|\mathsf{comN}(X_i)| \geq 2$ and $\{u, h_i\} \subseteq \mathsf{comN}(X_i)$. Now, by considering the bound on sizes of sets that have at least two common neighbors, we obtain that $|X_i|$ is also bounded by $f(t,d)$. We will now mark neighbors of vertices in $\widehat{Z}$ in X and will conclude that if X is sufficiently "large", then there exists an *irrelevant* vertex in X, that can be deleted.

For each vertex $h_i \in \widehat{Z}$ we observed above that $X_i = N(h_i) \cap X$ is bounded by $f(t,d)$. Therefore, for each $h_i \in \widehat{Z}$, we mark at most $f(t,d)$ vertices in X. By our observation, in phase one, $|N(S)|$ is bounded by $2t$ and therefore, we mark at most $2t \cdot f(t,d)$ vertices in X. Now, if the size of X is larger than $2t \cdot f(t,d)$, we can obtain a "good" vertex in X which is not adjacent to (does not dominate) any vertex in $\widehat{Z}$.

For a "large" sized set X (sufficiently larger than $2tf(t,d)$), if we delete a vertex, say $w \in X$, of smallest degree, then given any solution containing w, we can always obtain a *better or equally good solution* by replacing w with a *good* vertex in X. Employing the above discussion, by repeatedly applying a careful deletion procedure, we manage to bound the size of sets in A that have at least one common neighbor. By repeating these arguments, we can show that to bound the size of sets with at least 2 common neighbors, all we require is to bound size of sets with at least 3 common neighbors. Thus, inductively, we bound size of sets with $d-1$ common neighbors, by using the fact that the input graph is $K_{d,d}$-free. Thus, the algorithm starts for $d-1$ and applies a reduction rule to bound the size of sets with $d-1$ common neighbors. Once we apply the reduction rule for $d-1$ exhaustively, we apply for $d-2$ and by inductive application, we reach the one common neighbor case. To apply the reduction rule for $p \in [2, d-1]$, we assume that the reduction rules for $d-1$ common neighbor case have already been applied exhaustively. The bound on the size of the sets with one common neighbor also gives a bound on the degree of vertices in B by $f(t,d)$.

Finally in the third phase, the algorithm applies a reduction rule to delete all the vertices in A which are not amongst the first $g(t,d)$ high degree vertices in A, when sorted in non-increasing ordering of their degrees, which together with the upper bound and lower bound obtained on the degree of vertices in A, B obtained in phase one, gives the desired kernel size.

We now introduce our reduction rules. Our algorithm applies each reduction rule exhaustively in the order in which they are stated. We begin by stating some simple sanity check reduction rules.

Reduction Rule 5. *If $k < 0$, or $k = 0$ and $t \geq 1$, then return that (G, k, t) is a* no *instance instance of* Dec MaxRBDS.

The safeness of Reduction Rule 5 follows from the fact that the cardinality of any set cannot be negative and to dominate $t \geq 1$ vertices, a non empty set of vertices is required.

Reduction Rule 6. *If $k > 0$ and there exists a vertex $v \in A$ such that $\mathsf{deg}(v) \geq t$, then return that (G, k, t) is a* yes *instance instance of* Dec MaxRBDS.

The safeness of Reduction Rule 6 follows from the fact that the vertex v alone can dominate at least t vertices. When the above reduction rules are no longer applicable, we obtain the following fact.

Observation 7. *Suppose that S is a smallest solution to (G, k, t) of* Dec MaxRBDS. *Then, $|N(S)| \leq 2t$.*

Proof. Consider a vertex $u \in S$. If $|N(S \setminus \{u\})| \geq t$, then $S \setminus \{u\}$ is a smaller solution than S. Therefore, $|N(S \setminus \{u\})| < t$. Since Reduction Rule 6 is no longer applicable, we have $\mathsf{deg}(u) < t$. Hence, $|N(S)| \leq 2t$. □

The following reduction rule removes isolated vertices in G.

Reduction Rule 7. *If there exists $u \in V(G)$ such that $\mathsf{deg}(u) = 0$, then delete u from G. Return the resultant instance $(G = (A \setminus \{u\}, B, E), k, t)$ if $u \in A$, $(G = (A, B \setminus \{u\}, E), k, t)$ otherwise.*

The safeness of Reduction Rule 7 follows from the fact that such vertices cannot be in a minimal solution and cannot be dominated by any solution.

Next, we describe the reduction rules that help us bound the degree of vertices in the set B. For each $p \in [d-1]$, we introduce Reduction Rule 8.p. We apply Reduction Rule 8.p in the increasing order of p. That is, first apply Reduction Rule 8.1 exhaustively, and for each $p \in [d-1] \setminus \{1\}$, apply Reduction Rule 8.p only if Reduction Rule 8.$(p-1)$ has been applied exhaustively. We observe that after exhaustive application of the Reduction Rule 8.p, the resultant graph becomes $K_{t_p, d-p}$-free, where t_p is defined in as follows.

Definition 13. *For each $p \in [d-1]$, we define an integer t_p as follows:*

- *If $p = 1$, then $t_1 = 2t(d-1) + 1$, and*

- $t_p = 2t(t_{p-1}) + 1$, *otherwise.*

Observation 8. $t_{d-1} \leq (2t)^{d-1}d$

We recall that for a set $X \subseteq A$, by $\mathsf{comN}(X)$, we denote $\bigcap_{u \in X} N(u)$. For $X, S \subseteq A$, by $\mathsf{comN}(X, S)$, we denote $\mathsf{comN}(X) \cap N(S)$. We apply our reduction rule on a "large" sized subset of A where "large" is quantified by t_p defined in Definition 13.

Reduction Rule 8. *For each $p \in [d-1]$ we introduce* Reduction Rule 8.p *as follows. If there exists $X \subseteq A$ such that $|\mathsf{comN}(X)| = d - p$ and $|X| = t_p + 1$, then let v be a minimum degree vertex in X. Delete v from G and return the resultant instance (G', k, t), where $G' = (A \setminus \{v\}, B, E \setminus E_v)$.*

Lemma 17. *Reduction Rule 8 is safe.*

Proof. To show that the lemma holds, we will show that (G, k, t) is a yes instance instance of Dec MaxRBDS if and only if (G', k, t) is a yes instance instance of Dec MaxRBDS, for each $p \in [d-1]$. We prove the lemma by induction on p.

Base Case: $p = 1$. Suppose that $|X| = 2t(d-1) + 2$. In the forward direction, suppose that (G, k, t) is a yes instance instance of Dec MaxRBDS. Let S be its smallest solution. If $v \notin S$, then observe that S is also a solution to (G', k, t) of Dec MaxRBDS. Now, suppose that $v \in S$. Let $N(S) = \{h_1, \ldots, h_{|N(S)|}\}$. For each vertex $h_i \in N(S)$, let $X_i = X \cap N(h_i)$. Let $\widehat{Z} = N(S) \setminus \mathsf{comN}(X)$.

(1) By Observation 7, $|\widehat{Z}| \leq |N(S)| \leq 2t$.

(2) By the fact that G is a $K_{d,d}$-free graph, we obtain that for each vertex $h_i \in \widehat{Z}$, $|X_i| = |N(h_i) \cap X| \leq d - 1$. This is because $|\mathsf{comN}(X_i)| \geq d$ as $h_i \cup \mathsf{comN}(X) \subseteq \mathsf{comN}(X_i)$.

By (1) and (2), $|N(\widehat{Z}) \cap X| \leq 2t(d-1)$. This implies that there exists a vertex $w \in (X \setminus \{v\})$ such that $N(w) \cap \widehat{Z} = \emptyset$, as $|X \setminus \{v\}| = 2t(d-1) + 1$. We construct a set S' by replacing v in S by w, that is $S' = (S \setminus \{v\}) \cup \{w\}$. We have $N(S') = (N(S) \setminus N(v)) \cup N(w)$. Also, since $\mathsf{deg}(v) \leq \mathsf{deg}(w)$ and $N(v) \cap N(S) \supseteq N(w) \cap N(S)$, we have $|N(S')| \geq |N(S)| \geq t$. Since $S' \subseteq V(G')$, we have that S' is also a solution to (G', k, t) of Dec MaxRBDS. In the backward direction, suppose that (G', k, t) is a yes instance instance of Dec MaxRBDS and let S^* be its solution. Then, as G' is a subgraph of G, $N[S^*] \subseteq V(G)$. Thus, S^* is also a solution to (G, k, t) of Dec MaxRBDS.

Induction Hypothesis: Assume that Reduction Rule 8.q is safe for all $q \leq p-1$, $p \in [d-1]$.

Inductive Case: We have that $|X| = t_p + 1$. In the forward direction, suppose that (G, k, t) is a yes instance instance of Dec MaxRBDS. Let S be its smallest solution. If $v \notin S$, then observe that S is also a solution to (G', k, t) of Dec MaxRBDS. Now, suppose that $v \in S$.

As Reduction Rules 8.q are not applicable, where $q \leq p-1$, every set $Y \subseteq X$ such that $|\mathsf{comN}(Y)| \geq d-(p-1)$, satisfies that $|Y| \leq t_{p-1}$. In other words, for every set $Z \subseteq \widehat{Z}$ of size $d-(p-1)$, the set $X' \subseteq X$ such that $Z \subseteq \mathsf{comN}(X')$ has size at most t_{p-1}.

Let $N(S) = \{h_1, \ldots, h_t\}$. For each vertex $h_i \in N(S)$, let $X_i = X \cap N(h_i)$. Let $\widehat{Z} = N(S) \setminus \mathsf{comN}(X)$.

(1) By Observation 7, $|\widehat{Z}| \leq |N(S)| \leq 2t$.

(2) For each $h_i \in \widehat{Z}$, observe that $|\mathsf{comN}(X_i)| \geq d-p+1 = d-(p-1)$ as $\mathsf{comN}(X) \cup h_i \subseteq \mathsf{comN}(X_i)$. Which implies that $|X_i| \leq t_{p-1}$.

By (1) and (2) we obtain that $|N(\widehat{Z}) \cap X| \leq 2t(t_{p-1})$. Therefore, there exists a vertex $w \in (X \setminus \{v\})$ such that $N(w) \cap Z = \emptyset$, as $|X \setminus \{v\}| = 2t(t_{p-1}) + 1$. We construct a set S' by replacing v in S by w, i.e., $S' = (S \setminus \{v\}) \cup \{w\}$. We have that $N(S') = (N(S) \setminus N(v)) \cup N(w)$. Since $\mathsf{deg}(v) \leq \mathsf{deg}(w)$, and $\mathsf{comN}(X, S) = N(v) \cap N(S) \supseteq N(w) \cap N(S)$, we have $|N(S')| = |N(S)| - \mathsf{deg}(v) + \mathsf{deg}(w)$. Therefore, $|N(S')| \geq |N(S)| \geq t$. As $S' \subseteq V(G')$, we have that S' is also a solution to (G', k, t) of Dec MaxRBDS. In the backward direction, suppose that (G', k, t) is a yes instance instance of Dec MaxRBDS and let S^* be its solution. Then, as G' is a subgraph of G, $N[S^*] \subseteq V(G)$. Thus, S^* is also a solution to (G, k, t) of Dec MaxRBDS. This completes the proof. □

When Reduction Rule 8 is no longer applicable, we have that every set $X \subseteq A$ such that $|\mathsf{comN}(X)| \geq d-(d-1) = 1$, satisfies $|X| \leq t_{d-1}$. In other words, for every vertex $u \in B$, $|N(u)| \leq t_{d-1}$. We record this observation as follows.

Observation 9. *When Reduction Rules 5-8 are not applicable, then for every vertex $v \in B$, $\mathsf{deg}(v) \leq t_{d-1}$.*

Reduction Rule 9. *Let X be the set of first $\min\{n, 2t(t_{d-1})+1\}$ vertices in A when sorted by non-increasing order of their degrees. If there exists $v \in A \setminus X$, then delete v and return the resultant instance (G', k, t), where $G' = (A \setminus \{v\}, B, E \setminus E_v)$.*

Lemma 18. *Reduction Rule 9 is safe.*

Proof. To show that the lemma holds, we will show that (G, k, t) is a yes instance instance of Dec MaxRBDS if and only if (G', k, t) is a yes instance instance of Dec MaxRBDS. In the forward direction, suppose that (G, k, t) is a yes instance instance of Dec MaxRBDS and let S be its smallest solution. If $v \notin S$, then observe that S is also a solution to (G', k, t) of Dec MaxRBDS. Now, suppose that $v \in S$. Let $\widehat{Z} = N(S) \setminus \mathsf{comN}(X)$.

(1) By Observation 7, $|\widehat{Z}| \leq |N(S)| \leq 2t$.

(2) By Observation 9, we know that the degree of every vertex in B is bounded by t_{d-1}.

By (1) and (2), the number of vertices in X that have a neighbor in $\widehat{Z}$ is bounded by $2t(t_{d-1})$, that is $|N(\widehat{Z})| \leq 2t(t_{d-1})$. This implies that there exists a vertex $w \in (X \setminus \{v\})$ such that $N(w) \cap \widehat{Z} = \emptyset$, as $|X \setminus \{v\}| = 2t(t_{d-1}) + 1$. We construct a set S' by replacing v in S by w, i.e., $S' = (S \setminus \{v\}) \cup \{w\}$. We have $N(S') = (N(S) \setminus N(v)) \cup N(w)$. Since $\mathsf{deg}(v) \leq \mathsf{deg}(w)$, and $\mathsf{comN}(X, S) = N(v) \cap N(S) = N(w) \cap N(S) = S \setminus \widehat{Z}$, we have $|N(S')| = |N(S)| - \mathsf{deg}(v) + \mathsf{deg}(w)$. Therefore, $|N(S')| \geq |N(S)| \geq t$. As $S' \subseteq V(G')$, we have that S' is also a solution to (G', k, t) of Dec MaxRBDS. In the backward direction, suppose that (G', k, t) is a yes instance instance of Dec MaxRBDS and let S^* be its solution. Then as G' is a subgraph of G, $N[S^*] \subseteq V(G)$. Thus, S^* is also a solution to (G, k, t) of Dec MaxRBDS. This completes the proof. □

Observe that Reduction Rules 5-7 and 9 can be applied in polynomial time. Reduction Rule 8 can be applied in $n^{\mathcal{O}(d)}$ time. Each of our reduction rules is applicable only polynomial many times. Hence, the kernelization algorithm runs in polynomial time. Each of our reduction rule is safe. When Reduction Rule 9 is no longer applicable, the size of the set A is bounded by $2t(t_{d-1})+1$, the degree of each vertex in A is bounded by t, and there are no isolated vertices in U. Therefore, the number of vertices in G is bounded by $t(2t((2t)^{d-1}d)+1)$ (that is $\mathcal{O}(t^{d+1}d)$) and by Observation 8, we obtain the following theorem.

Theorem 14. Dec MaxRBDS *in $K_{d,d}$-free graphs admits a kernel of size $\mathcal{O}(t^{d+1}d)$.*

Part III: Optimizing objects to satisfy constraints

Chapter 10

PAS for Coverage parameterized by solution size

In contrast to the previous chapters, where we aimed to approximate the coverage requirements (with a smallest sized solution), here, we present a scheme that approximates (with an additive factor) the solution size while satisfying the coverage requirements exactly. Specifically, we give a PAS parameterized by k for Max Red Blue Dominating Set with Covering Constraint and its generalization with fairness constraints in Chapters 10.2 and 10.3 respectively. We also give a hardness of approximation result for the same in Chapter 10.4.

10.1 Overview of the PAS

When $\varepsilon \geq \frac{1}{k}$, we invoke the one-additive approximation algorithm, whereas for $\varepsilon < \frac{1}{k}$, we optimally solve by trying all possible sets of size at most k in $n^{\mathcal{O}(\frac{1}{\varepsilon})}$ time which is allowed in a PAS. We use the following strategy for the additive approximation algorithm. For each $i \in [n]$, we invoke the EPAS for MaxRBDS with $\varepsilon = \frac{1}{4i}$. Note that for $i = k$, this will definitely return us a set $S' \subseteq A$ of size at most k such that $|N(S)| \geq t(1 - \frac{1}{4k})$ (this may even happen for $i < k$). The idea is then to try and use the leverage of an extra vertex to dominate (a vertex u *dominates* a vertex v if $v \in N(u)$) the additional $\frac{t}{4k}$ vertices in B. In the case when we cannot find that *additional vertex*, we are able to bound the number of vertices in A with *sufficiently* high degree using the properties of $K_{d,d}$-free graphs, in particular, using the combinatorial lemma (Lemma 6). Furthermore, we argue

that every set $X \subseteq A$ of size at most k, such that $|N(X)| \geq t$, contains at least one such high degree vertex. Then, we branch on the choices for this vertex v, and recursively solve the problem. In any iteration, if we are not able to return a set $S^\star \subseteq A$ of size $k+1$ with $|N(S^\star)| \geq t$, we will be able to find a new vertex of the optimal solution and hence the depth of the recursion tree is bounded by k.

10.2 A one-additive approximation for Max Red Blue Dominating Set with Covering Constraint on $K_{d,d}$-free instances

We present a one-additive approximation algorithm running in FPT time for Max Red Blue Dominating Set with Covering Constraint on $K_{d,d}$-free instances and state our main result below.

Theorem 15. *There exists an algorithm for* Max RBDS CovC*, that given a $K_{d,d}$-free bipartite graph G, with $V(G) = A \uplus B$, an integer k, runs in time $(kd)^{\mathcal{O}(kd)} n^{\mathcal{O}(1)}$, and returns a set $S \subseteq A$ of size at most $k+1$ such that $|N(S)| \geq t$, where $k = \mathsf{OPT}_t(G)$.*

Recall that an input to Max RBDS CovC consists of a bipartite graph G, with $V(G) = A \uplus B$, and an integer t, and the objective is to find a set $S \subseteq A$ of size k such that $|N(S)| \geq t$, where $k = \mathsf{OPT}_t(G)$. We call a set $A' \subseteq A$ of size k with $|N(A')| \geq t$ as a *t-optimal* solution. For our approximation algorithm, we use EPAS algorithm for MaxRBDS (Theorem 3), the polynomial kernel for Dec MaxRBDS (Theorem 14) in conjunction with the following lemma, which states that if we take sufficiently many high degree vertices, then either one of them goes into the optimum solution or we can find a vertex from this set that together with the approximate solution returned by the FPT-AS algorithm for MaxRBDS covers the requisite number of vertices.

Lemma 19. *Let G be a $K_{d,d}$-free bipartite graph with $V(G) = A \uplus B$, and let $\ell \leq k$ and t' be two positive integers. Let $S' \subseteq A$ be an ℓ sized set such that $|N(S')| \geq t'(1 - \frac{1}{4\ell})$, where $t' > 8\ell^4 d$, and H be the set of $\ell(d-1)(4\ell^2)^d + 1$ highest degree vertices from A. For any $S \subseteq A$ of size ℓ with $|N(S)| \geq t'$, either $S \cap H \neq \emptyset$ or there exists a vertex $x \in H$ such that $|N(\{x\} \cup S')| \geq t'$.*

Proof. Suppose a lowest degree vertex in H has degree less than $\frac{t'}{\ell+1}$. Since S is a

set of size ℓ, if $S \cap H = \emptyset$, then all the vertices in $V(G) \setminus H$ have degree less than $\frac{t'}{\ell+1}$ and hence $N(S) < \ell \cdot \frac{t'}{\ell+1} < t'$ which contradicts the fact that $|N(S)| \geq t'$. From here onwards, we assume that every vertex in H has degree at least $\frac{t'}{\ell+1}$. We will prove the above lemma using a contradiction. Towards a contradiction, assume that for every $y \in H$, let $|N(\{y\} \cup S')| < t'$. But, $|N(y)| \geq \frac{t'}{\ell+1}$, and $|N(S')| \geq t'(1 - \frac{1}{4\ell})$. Hence,

$$\begin{aligned} |N(y) \cap N(S')| &= |N(S')| + |N(y)| - |N(\{y\} \cup S')| && (10.1) \\ &\geq t' \left(1 - \frac{1}{4\ell}\right) + \frac{t'}{\ell+1} - t' && (10.2) \\ &= t' \left(\frac{1}{\ell+1} - \frac{1}{4\ell}\right) && (10.3) \\ &\geq \frac{t'}{2\ell}. && (10.4) \end{aligned}$$

From pigeonhole principle, for every y, there exists a vertex $v_i \in S'$ such that

$$|N(y) \cap N(v_i)| \geq \frac{|N(y) \cap N(S')|}{|S|(=\ell)} \geq \frac{t'}{2\ell^2}.$$

Again using pigeonhole principle, we can conclude that $\exists v \in S'$ such that there are at least $\frac{|H|}{\ell}$ many vertices $y \in H$, with $|N(y) \cap N(v)| \geq \frac{t'}{2\ell^2}$. We denote all these vertices by H_v. Now we construct a bipartite graph $G_v = G[H_v \cup N(v)]$. Notice that we can always assume $|N(v)| \leq t'$ (otherwise for any $x \in H$ we simply return $\{x\} \cup S'$ as the desired t'-optimal solution). We set $\beta = 2\ell^2$. If $\frac{|B|}{2\beta} \leq d$, then $|N(v)| = |B| \leq 2\beta d \leq 4\ell^2 d$. As, $|N(v)| \geq |N(v) \cap N(y)| \geq \frac{t'}{2\ell^2}$, we get that $t' \leq 8\ell^4 d$, which is a contradiction. Thus, we can assume that $\frac{|B|}{2\beta} > d$. This implies that $H_v \subseteq \mathsf{AHD}_\beta^{G_v}(N(v))$ (as defined in Equation 5.5), since every vertex in H_v has degree at least $\frac{t'}{\beta} \geq d$. By applying Lemma 6, we get $|H_v| \leq (d-1)(4\ell^2)^d$. But this contradicts the fact that $|H| \geq \ell(d-1)(4\ell^2)^d + 1$. □

With the above lemma in our hand, we give an FPT approximation algorithm for Max RBDS CovC. We will use $g(\ell, d)$ as a place holder for $\ell(d-1)(\ell^2)^{d-1} + 1$. We refer to Figure 9 for a detailed description of the algorithm. The algorithm in Figure 9 invokes the algorithm in Figure 8 for each value of i to find a solution of size $i + 1$.

Proof of Theorem 15. An input to our algorithm consists of a $K_{d,d}$-free bipartite graph $G = (A, B, E)$ and an integer t. We call the algorithm APX-k-RBDS which

Algorithm 8 APX-check-RBDS$(G = (A, B, E), t, i)$

1: For $i = 0$, if there exists $x \in A$ such that $|N(x)| \geq t$, return $\{x\}$, else just return.
2: **if** $t \leq 4i^2d$ **then**
3: Reduce the size of the graph to $\mathcal{O}(t^{d+1}d)$, using Theorem 14. Find an optimal t-solution S by brute-force, and return S if and only if $|S| \leq i + 1$.
4: **else if** $|A| \leq g(i, d)$ **then**
5: Find an optimal t-solution S (check all subsets of A) in $\binom{|A|}{i+1}$ time and return S if and only if $|S| \leq i + 1$.
6: **else**
7: Let S'=APX-RBDS$(G = (A, B, E), i, \varepsilon = \frac{1}{4i})$.
8: Let $H \subseteq A$ be a set of $g(i, d)$ highest degree vertices from A.
9: **if** for any $x \in H$, $|N(\{x\} \cup S')| \geq t$ **then**
10: **return** $\{x\} \cup S'$.
11: **else if** For any $x \in H$, if APX-check-RBDS$(G - N_G[x], i - 1, t - |N_G(x)|)$ returns a set S **then**
12: **return** $S \cup \{x\}$, if $|N(S \cup \{x\})| \geq t$.
13: **end if**
14: **end if**

Algorithm 9 APX-k-RBDS$(G = (A, B, E), t)$

1: **for** each value of i from 0 to n in increasing order **do**
2: **if** APX-check-RBDS$(G = (A, B, E), t, i)$ returns a set S with $|N(S)| \geq t$ **then**
3: **return** S and exit.
4: **end if**
5: **end for**

in turn calls the recursive algorithm APX-check-RBDS$(G = (A, B, E), t, i)$ for $i = 0$ to n. Our algorithm terminates when APX-k-RBDS returns a solution set for the smallest value of i. In the iteration APX-check-RBDS$(G = (A, B, E), t, OPT_t(G))$, the algorithm at any step either returns a one-additive approximate solution (Steps 2, 3, and 6) or is able to find a vertex of t-optimal solution (Step 7). For the second case we remove this vertex as well as all its neighbors and proceed to the next step of the recursion.

Approximation Ratio. We show that the algorithm APX-check-RBDS returns a set $S \subseteq A$, of size at most $k + 1$, such that $|N(S)| \geq t$ by induction on k.

Base Case. When $k = 0$, our algorithm returns a set $\{x\}$ if and only if $|N(x)| \geq t$ and thus the statement holds.

Inductive Case. Here we have 4 subcases.

Case (i): The first case is when $t \leq 4k^2d$.

Case (ii): The second case is when $|A| \leq g(k,d)$.

Case (iii): The third case is when there exists $x \in H$, $|N(\{x\} \cup S')| \geq t$.

For the first three cases the algorithm only returns a set S if and only if $|S| \leq k+1$ and $|N(S)| \geq t$. So the correctness of the inductive step is evident.

Case (iv): In this case $t > 4k^2d$, $|A| > g(k,d)$ and for all $x \in H$, $|N(\{x\} \cup S')| < t$. Due to Lemma 19, for any $S \subseteq A$ of size k with $|N(S)| \geq t$, $S \cap H \neq \emptyset$. So there exists a vertex $v \in H \cap S$. We claim that $S = \{v\} \cup$APX-check-RBDS$(G - N_G[v], k-1, t-|N_G(v))|$ is an at most $k+1$ sized set with $|N(S)| \geq t$. From induction $(G - N_G[v], k-1, t-|N_G(v))|$ returns us a set S' of size at most k such that $|N(S')| \geq t-|N_G(v))|$ in $G-N_G[v]$, since $OPT(G-N_G[v], t-|N_G(v))|) = k-1$. Adding v to S', we also dominate all the vertices in $N_G[v]$. Hence $\{v\} \cup S'$ is a set of size at most $k+1$ such that with $|N(\{v\} \cup S')| \geq t$.

Running time. The running time of the algorithm APX-check-RBDS is governed by the following set of recurrence equations.

1. $T(k) \leq g(k,d) \cdot T(k-1) + (kd)^{\mathcal{O}(kd)} + g(k,d)^{k+1} n^{\mathcal{O}(1)} + \left(\frac{k}{\varepsilon}\right)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ (where $\varepsilon = \frac{1}{4k}$).
2. $T(0) = n^{\mathcal{O}(1)}$.

Notice that Steps 2, 3 and 4 of the algorithm take $(kd)^{\mathcal{O}(kd)}$, $g(k,d)^{k+1}n^{\mathcal{O}(1)}$ and $(4k)^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time, respectively, while all other steps take $n^{\mathcal{O}(1)}$ time. And the above recurrence equations solve to $T(k) \leq g(k,d)^k((kd)^{\mathcal{O}(kd)} + g(k,d)^{k+1} + (4k)^{\mathcal{O}(k)})n^{\mathcal{O}(1)}$. Thus, $T(k) \leq (kd)^{\mathcal{O}(kd)}n^{\mathcal{O}(1)}$. The running time of algorithm APX-k-RBDS is bounded by

$$f(k,d) = \sum_{i=0}^{k} T(i) \leq k((kd)^{\mathcal{O}(kd)}n^{\mathcal{O}(1)}) = (kd)^{\mathcal{O}(kd)}n^{\mathcal{O}(1)}.$$

The smallest value of i for which our algorithm returns an at most $i+1$ size set S (a solution), is either $k-1$ or k. We show that S is a 1-additive approximation. If $S = S_1 =$APX-check-RBDS$(G, k-1, t)$ and $S_2 =$APX-check-RBDS(G, k, t), then $|S_1| \leq |S_2|$. Since, S_2 is a 1-additive approximation solution, so is S_1. The correctness when $S =$APX-check-RBDS(G, k, t) is pronounced. □

10.3 An r-additive approximation algorithm for Partition Color Constrained Dominating Set on $K_{d,d}$-free graphs

We generalize the result of Section 10.2 by designing an r-additive FPT approximation algorithm for Partition Color Constrained Dominating Set (or PCCDS) on $K_{d,d}$-free bipartite graphs. We recall the formal definition of PCCDS here.

> Partition Color Constrained Dominating Set (PCCDS)
> **Input:** An instance $\mathcal{I} = (G, r, f, k, t)$, where $G = (A \uplus B, E)$ is a bipartite graph, $f : B \to [r]$ is a surjective function, a non-negative integer k, and for each color $j \in [r]$, a *coverage requirement* $t_j \geq 0$.
> Here, $B_j \coloneqq \{\mathfrak{s} \in B : f(\mathfrak{s}) = j\}$ is a set of vertices of *color* j, for each $j \in [r]$.
> **Parameter:** k
> **Question:** Does there exist a subset $S \subseteq A$, such that (1) $|S| \leq k$, and (2) for each $j \in [r]$, $|N_j(S)| \geq t(j)$? Here, $N_j(S) \subseteq B_j$ is the set of vertices of color $j \in [r]$ that are adjacent to at least one vertex in S.

Given an instance $(G = (A \uplus B, E), r, f, k, t)$ of the PCCDS problem, we return a solution $S \subseteq A$ of size at most r larger than any optimal solution while satisfying the coverage requirement $|N_j(S)| \geq t_j$, for each color j. To design this algorithm, we crucially use Lemma 19 from the previous section.

The lemma states that in a unicolored $K_{d,d}$-free bipartite graph $G = (A \uplus B, E)$, the set H of the first $g(k,d)$ many highest degree vertices satisfies certain properties, where k is the minimum size of a subset S of A such that $|N_j(S)| \geq t_j$, for each color $j \in [r]$ and $g(k,d) = k(d-1)(4k^2)^{d-1}+1$. Specifically, for any optimal solution O of PCCDS (unicolored version) on G, either $O \cap H = \emptyset$ or there exists a vertex $y \in H$ which together with a *reasonably good* approximate solution S forms a one-additive approximate solution S'. Note that the input graph in the lemma has to be unicolored. In our algorithm, we are able to remove this restriction (input graph is r-colored from unicolored) while achieveing an r-additive approximation instead of a one-additive approximation. We state our main theorem below.

For any optimal solution O of Partition Color Constrained Dominating Set on G, either $O \cap H \neq \emptyset$ or there exists a vertex $y \in H$ such that $|N_j(S' \cup \{y\})| \geq$

t_j for each $j \in [r]$ for an approximate solution S'. We formally restate the lemma here for convenience.

Algorithm 10 ADDITIVE-PCCDS($\mathcal{I} = (G = (A \uplus B, E), [r], f, t)$)

1: **for** each i from 0 to $|A|$ **do**
2: **if** CHECK-ADDITIVE-PCCDS($G, [r], f, t, i$) returns a set S with $|N_j(S)| \geq t_j$, for each $j \in [r]$ **then**
3: **return** S.
4: **end if**
5: **end for**

Theorem 16. *There exists an algorithm for* PARTITION COLOR CONSTRAINED DOMINATING SET *that takes as input a bipartite graph $G = (A \uplus B, E)$, that is $K_{d,d}$ free, with $f : B \to [r]$, the coverage requirement t_j and returns a set S of size at most $k + r$ such that $|N_j(S)| \geq t_j$, for each color $j \in [r]$. Here, k is the minimum value of $|S|$ such that $|N_j(S)| \geq t_j$, for each color $j \in [r]$. The algorithm runs in time $(2^{\mathcal{O}(k^5 r^2 d \log r)} + (kd)^{\mathcal{O}(kd)}) \cdot (n)^{\mathcal{O}(1)}$.*

Proof. We call the algorithm ADDITIVE-PCCDS which takes as input a $K_{d,d}$-free bipartite graph $G = (A \uplus B, E)$ with $f : B \to [r]$ and the coverage requirement t_j for each color $j \in [r]$. ADDITIVE-PCCDS returns a set S of size at most $k + r$ such that $|N_j(S)| \geq t_j$, for each $j \in [r]$ if there is a solution of size k in G. Here, k is the minimum size of a set in A that dominates at least t_j elements of color j in B.

ADDITIVE-PCCDS in turn calls the algorithm CHECK-ADDITIVE-PCCDS for each value of i from 0 to $|A|$. For a particular value of i, CHECK-ADDITIVE-PCCDS returns a set S of size at most $i + r$ such that $|N_j(S)| \geq t_j$, for each $j \in [r]$, if there is a solution of size i. Hence, there exists an $i \leq k$ for which it returns an r-additive approximate solution.

Additive approximation factor: We prove a bound on the additive approximate solution returned by the CHECK-ADDITIVE-PCCDS algorithm (when it has a solution of size at most i) using induction on i.

Base Case: For $i = 0$, CHECK-ADDITIVE-PCCDS checks if the coverage requirement for each color is 0. If so, it returns a solution of size exactly r. Otherwise, no solution is possible and it returns a statement stating the same.

Inductive Case: For $i > 0$, we have the following three cases.

Case 1.[$|A| < g(i, d)$] In this case, the algorithm returns a set only if it is of size

Algorithm 11 CHECK-ADDITIVE-PCCDS($\mathcal{I} = (G = (A \uplus B, E), [r], f, t, i)$)

Define $S = \emptyset$.
if $i = 0$ **then**
  if for each $j \in [r]$, $t_j = 0$ **then**
    return $\emptyset$.
  else
    for each $j \in [r]$ with $t_j > 0$, check if there exists a vertex $x_j \in A$ such that $|N_j(x_j)| \geq t_j$ **do**
      $S = S \cup \{x_j\}$
    end for
  end if
end if
if $|A| \leq g(i, d)$ **then**
  check all subsets A' of A and **return** A' if and only if $|A'| \leq i + r$ and $|N_j(A')| \geq t_j$ for each $j \in [r]$.
end if
Let $\hat{S}$ =$K_{d,d}$-FREE-PCCDS($G', [r], f, t', i, \frac{1}{4i}$), where $G' = G - \cup_{j:t_j<4i^2d} B_j$ and $t'_j = 0$ for each j when $t_j < 4i^2d$. ($\hat{S}$ is a $(1 - \frac{1}{4i})$-approximate solution.) (Note that we call the derandomized version of $K_{d,d}$-FREE-PCCDS here)
for each $j \in [r]$ **do**
  Construct the set H_j of $g(i, d)$ highest degree vertices from $A \setminus \hat{S}$ with respect to the color j.
end for
for each $j \in [r]$ **do**
  if for all $y \in H_j$, $|N_j(\hat{S} \cup \{y\})| < t_j$ **then**
    for all $y \in H_j \cup \hat{S}$ **do**
      Let S' be the set returned by CHECK-ADDITIVE-PCCDS($G' - N_j(y), [r], f, t', i - 1$) (where $G' = G - \cup_{l\in[r]:t_l<4i^2d} B_l$ and $t'_p = 0$ when $t_p < 4i^2d$ and $t'_j = t_j - |N_j(y)|$)
      return $S' \cup \{y\}$ if $|N_j(S' \cup \{y\})| \geq t_j$
    end for
  else there exists a $y \in H_j$ such that $|N_j(\hat{S} \cup \{y\})| \geq t_j$
    Include y in S.
  end if
end for
return $\hat{S} \cup S$.

at most $i + r$ and it satisfies the coverage requirement for each color. Moreover, the returned solution is not only an r-additive approximation but also optimal.

Case 2.[For each $j \in [r]$ with $t_j > 4i^2d$, there exists a $y \in H_j$ such that $|N_j(\hat{S} \cup y)| \geq t_j$.] In this case, CHECK-ADDITIVE-PCCDS checks if there is a vertex for each color which when added to the $(1 - \frac{1}{4i})$-approximate solution $\hat{S}$, returned by $K_{d,d}$-FREE-PCCDS, satisfies the coverage requirement for the corresponding color. Since,

$|\hat{S}| \leq i$ and exactly r vertices are added to it, we get an r-additive solution.

Case 3.[For some $j \in [r]$ with $t_j > 4i^2d$, there is no $y \in H_j$ such that $|N_j(\hat{S} \cup y)| \geq t_j$.] In this case, because of Lemma **??** from [**?**], we know that for any optimal solution O ($|O| \leq i$ and $|N_j(O)| \geq t_j$ for each $j \in [r]$), $O \cap (H_j \cup \hat{S}) \neq \emptyset$. Let $y \in O \cap (H_j \cup \hat{S})$ and $S' =$ CHECK-ADDITIVE-PCCDS$(G' - N_j(y), [r], f, t', i-1)$ where $G' = G - \cup_{l \in [r]: t_l < 4i^2d} B_l$ and $t'_p = 0$ when $t_p < 4i^2d$ and $t'_j = t_j - |N_j(y)|$. The algorithm returns $S' \cup \{y\}$, if this set satisfies the coverage requirement for the color j. By induction hypothesis, S' is a $(i-1+r)$-sized set that satisfies the coverage requirements for all colors except j and covers at least $t_j - |N_j(y)|$ many vertices in B for color j. Thus, when we add y to S', we get a $(i+r)$-sized set (an r-additive approximation) with $|N_j(S' \cup \{y\})| \geq t_j$ for all $j \in [r]$.

(Notice that the scenario where there exists a color $j \in [r]$ such that $t_j < 4i^2d$ is handled by the $K_{d,d}$-FREE-PCCDS algorithm itself, i.e., the approximate solution $\hat{S}$ returned by the algorithm satisfies the coverage requirement t_j.)

Running Time: We first determine the running time of the recursive algorithm CHECK-ADDITIVE-PCCDS.

- Steps $2-10$ and $15-17$ take $n^{\mathcal{O}(1)}$ time.
- Steps $11-13$ take $g(k,d)n^{\mathcal{O}(1)}$ time.
- Step 14 takes $2^{\mathcal{O}(k^5r^2d\log r)} \cdot (n)^{\mathcal{O}(1)}$ time.
- Steps $18-27$ take $(g(k,d)+k)T(k-1) + n^{\mathcal{O}(1)}$ time.

As a result, we get the following set of recurrence equations.

$$T(k) = n^{\mathcal{O}(1)} + g(k,d)n^{\mathcal{O}(1)} + 2^{\mathcal{O}(k^5r^2d\log r)} \cdot (n)^{\mathcal{O}(1)} + (g(k,d)+k)T(k-1)$$

and $T(0) = n^{\mathcal{O}(1)}$ which results in a running time of

$$T(k) \leq ((2^{\mathcal{O}(k^5r^2d\log r)} + (k(d-1)(4k^2)^{d-1} + k)^k)(n)^{\mathcal{O}(1)}$$

The running time for ADDITIVE-PCCDS is at most

$$kT(k) \leq (2^{\mathcal{O}(k^5r^2d\log r)} + (kd)^{\mathcal{O}(kd)})(n)^{\mathcal{O}(1)}.$$

This concludes the running time analysis. □

10.4 Hardness of approximation for Partition Color Constrained Dominating Set

We prove that Partition Color Constrained Dominating Set does not admit any algorithm on $K_{d,d}$-free bipartite graphs running in time $f(k,r,d)\cdot n^{\mathcal{O}(1)}$ that returns an additive $(r-1)$ approximate solution for every $r>0$. To show this, we give a reduction from the Partial Vertex Cover problem where we are given a graph G and an integer k and the objective is to find a vertex subset S of G of size at most k that maximizes the number of edges with one end point in S. It is well known to be W[1]-hard parameterized by the solution size (k) [72].

Construction: Given an instance (G',k,t^*) of the Partial Vertex Cover problem, we create an equivalent instance of PCCDS as follows. First we create a bipartite graph G where $V(G)=A\uplus B$ with $A=V(G')$ and $B=E(G')$ and there is an edge between two vertices $x\in A$ and $e\in B$ if and only if x is an endpoint of the edge e in G'. Note that there is a k-sized set $S\subseteq V(G')$ covering t^* edges in the Partial Vertex Cover instance (G',k,t^*) if and only if there is a k-sized set $S\subseteq A$ dominating t^* elements in B. We create r many disjoint copies of the graph G, named $\{G_1,\ldots,G_r\}$ with $V(G_i)=A_i\cup B_i$ for each $i\in[r]$ and construct a new graph $H=G_1\uplus G_2,\ldots\uplus G_r$ with $V(H)=A\cup B$, where $A=A_1\uplus\ldots\uplus A_r$ and $B=B_1\uplus\ldots\uplus B_r$. The coloring function f maps each vertex in B_i to a color i. We set $t_i=t^*$ for each $i\in[r]$. Note that H is $K_{2,2}$-free.

Claim 5. *(G',k,t^*) is a yes instance of Partial Vertex Cover if and only if $(H,r,f,t,rk+r-1)$ is a yes instance of PCCDS.*

Proof. In the forward direction, suppose (G',k,t^*) is a yes instance of Partial Vertex Cover problem. Let $S\subseteq V(G')$ be a solution of size at most k covering at least t^* edges. Equivalently, in the bipartite graph G constructed from G', we have the set $S\subseteq A$ dominating at least t^* vertices in B. Now we create a solution for the PCCDS instance as follows. In H, by taking the exact copies of the corresponding vertices of S in each G_i, we construct a set S^* of size at most rk, i.e., $S^*=S_1\uplus\ldots\uplus S_r$. This indeed is a solution (of size rk) since, for each $i\in[r]$, S_i dominates at least t^* (i-colored) vertices from B_i. Hence the PCCDS instance $(H,r,f,t,rk+r-1)$ has a solution of size at most $rk\leq rk+r-1$.

In the reverse direction, let S^* be a solution of size at most $rk+r-1$ for the given PCCDS instance, i.e., $S^*\subseteq A$ is of size at most $rk+r-1$ that dominates/covers t^* many vertices for each color $i\in[r]$. Then from the pigeonhole principle there exists

$i \in [r]$ such that $|S^* \cap A_i| \leq k$. But $S^* \cap A_i$ are the only vertices in A that dominate the t^* vertices of color i in B. This directly implies that t^* many edges in G' can indeed be covered by k vertices and equivalently (G', k, t^*) is a yes instance. □

By setting $r = g(k)$ for any computable function g in the above construction (where $d < 2$), the W[1]-hardness (w.r.t the solution size k) of Partial Vertex Cover problem immediately implies the following theorem.

Theorem 17. PCCDS *on $K_{d,d}$-free bipartite graphs does not admit any $(r-1)$ additive approximate algorithm running in time $f(k,r,d) \cdot n^{\mathcal{O}(1)}$ unless FPT = W[1].*

Part IV: Multi-criteria Lossy Kernelization

Chapter 11

Framework for lossy kernels for Multi-criteria optimization problems

This chapter formalizes the notion of parameterized multi-criteria optimization problems and develops a framework for lossy kernelization in this setting. Thus far, our focus has been on parameterized approximation schemes for partial cover problems and their variants. We now turn to the design of lossy kernels for these problems, motivated by the fundamental equivalence between parameterized approximation schemes and lossy kernelization—the progress of one inherently informs the other. The introduction of additional fairness and matroid constraints naturally necessitates a multi-criteria perspective, leading us to define multi-criteria optimization problems and to establish a corresponding theory of lossy kernels for them.

We begin by defining what we mean by a parameterized multi-criteria optimization problem.

Definition 14 (**Parameterized Multi-Criteria Optimization Problem**). *A parameterized multi-criteria optimization problem* Π *is a set of* $r + 1$ *computable functions where* $r = \mathcal{O}(1)$*. The functions are described as follows.*

1. *A main optimization function:*

$$w \colon \Sigma^\star \times \mathbb{N} \times \Sigma^\star \to \mathbb{R} \cup \{\pm\infty\}.$$

2. A set of r constraint functions:

$$\forall i \in [r],\ \ell^i \colon \Sigma^\star \times \mathbb{N} \times \Sigma^\star \to \mathbb{R} \cup \{\pm\infty\}.$$

The instances of a parameterized multi-criteria optimization problem Π are tuples

$$(I, k, t_1, \ldots, t_r) \in \Sigma^\star \times \mathbb{N}^{r+1}$$

and a solution to $(I, k, t_1, \ldots, t_r)$, which is a string $s \in \Sigma^\star$ such that it satisfies the following conditions.

1. $|s| \leq |I| + k + \sum_{i \in [r]} t_i$
2. $\forall i \in [r]$, $\ell^i(I, k, s) \geq (\leq) t_i$

Here, k is called the parameter. The value of the solution s is denoted by $w(I, k, s)$. The algorithmic task is to find a solution with the optimal value, which is either the minimum or maximum among all solutions, depending on whether it is a minimization or a maximization problem, respectively.

Remark 3. *Optimization problems that seek approximation of the type $\in (1 \pm \epsilon)c_i$ can be handled by two constraints of $\geq, \leq$.*

Definition 15 (Optimum Value). *For a parameterized multi-criteria maximization problem Π, the optimum value of an instance*

$$(I, k, t_1, \ldots, t_r) \in \Sigma^\star \times \mathbb{N}^{r+1}$$

is given by

$$\mathsf{OPT}_\Pi(I, k, t_1, \ldots, t_r) = \max_{s \text{ is a solution}} w(I, k, s)$$

Similarly for a parameterized multi-criteria minimization problem Π, the optimum value of an instance

$$(I, k, t_1, \ldots, t_r) \in \Sigma^\star \times \mathbb{N}^{r+1}$$

is given by

$$\mathsf{OPT}_\Pi(I, k, t_1, \ldots, t_r) = \min_{s \text{ is a solution}} w(I, k, s).$$

For an instance $(I, k, t_1, \ldots, t_r)$ of a parameterized multi-criteria optimization problem Π, an optimal solution is a solution s that achieves the optimum value, i.e., $w(I, k, s) = \mathsf{OPT}_\Pi(I, k, t_1, \ldots, t_r)$.

Notice that $\mathsf{OPT}_\Pi(I, k, t_1, \ldots, t_r)$ is well-defined as the set of solutions over which we are optimizing is finite.

Definition 16 (**solving** Π)**.** *Let* Π *be a parameterized multi-criteria optimization problem. An algorithm for* Π *is an algorithm that takes as input an instance* $(I, k, t_1, \ldots, t_r)$*, outputs a solution* s *and halts. The algorithm solves* Π *if, for every instance* $(I, k, t_1, \ldots, t_r)$*, the solution* s *output by it is optimal for* $(I, k, t_1, \ldots, t_r)$*. We say that a parameterized multi-criteria optimization problem,* Π*, is* decidable *if there exists an algorithm that solves* Π *.*

Definition 17 (**FPT multi-criteria optimization problem**)**.** *A parameterized multi-criteria optimization problem* Π *is fixed parameter tractable (FPT) if there is an algorithm that solves it and the running time of the algorithm on any instance, say* $(I, k, t_1, \ldots, t_r)$*, of size* n *with parameter* k *is upper bounded by* $f(k)n^{\mathcal{O}(1)}$ *for a computable function* f*.*

Definition 18 (**Polynomial time preprocessing algorithm for a parameterized multi-criteria optimization problem**)**.** *Polynomial time preprocessing algorithm* $\mathcal{A}$ *for a parameterized multi-criteria optimization problem* Π *is a pair of polynomial time algorithms. The first one is called the* ***reduction algorithm*** *and computes a map* $\mathcal{R}_\mathcal{A} \colon \Sigma^\star \times \mathbb{N}^{r+1} \to \Sigma^\star \times \mathbb{N}^{r+1}$*. Given as input an instance* $(I, k, t_1, \ldots, t_r)$ *of* Π*, the reduction algorithm outputs another instance* $(I', k', t'_1, \ldots, t'_r) = \mathcal{R}_\mathcal{A}(I, k, t_1, \ldots, t_r)$*.*

The second algorithm is the ***solution-lifting algorithm****. It takes as input an instance* $(I, k, t_1, \ldots, t_r) \in \Sigma^\star \times \mathbb{N}^{r+1}$ *of* Π*, the reduced instance* $(I', k', t'_1, \ldots, t'_r)$ *produced by the reduction algorithm, and a solution* s' *to* $(I', k', t'_1, \ldots, t'_r)$*. It runs in time polynomial in the size of its input and outputs a solution* s *to* $(I, k, t_1, \ldots, t_r)$*. Finally, if* s' *is an optimal solution to* $(I', k', t'_1, \ldots, t'_r)$*, then* s *is an optimal solution to* $(I, k, t_1, \ldots, t_r)$*.*

Definition 19 (**Kernel for a parameterized multi-criteria optimization problem**)**.** *A* kernelization *(or a* kernel*) for a parameterized multi-criteria optimization problem* Π *is a polynomial time preprocessing algorithm* $\mathcal{A}$ *such that aliteraliter*

$$size_\mathcal{A}(k) = \sup\{|I'| + k' + t'_1 + \cdots + t'_r \colon (I', k', t'_1, \ldots, t'_r) = \mathcal{R}_\mathcal{A}(I, k, t_1, \ldots, t_r), \\ I \in \Sigma^\star, \forall i \in [r], t_i \in \mathbb{N}\}$$

is upper bounded by a computable function $g \colon \mathbb{N} \to \mathbb{N}$ *of the parameter* k*.*

Proposition 8. *A decidable parameterized multi-criteria optimization problem is FPT if and only if it admits a kernel.*

Proof. First, we prove the backward direction. Given an instance $(I, k, t_1, \ldots, t_r)$, one may run the reduction algorithm to obtain a new instance $(I', k', t'_1, \ldots, t'_r)$ of size bounded by a computable function of k, say g. Since $(I', k', t'_1, \ldots, t'_r)$ has bounded size and Π is decidable, an optimal solution s' can be found in time at most $g'(k)$ for some computable function g'. Finally, one can use the solution-lifting algorithm to obtain an optimal solution s to $(I, k, t_1, \ldots, t_r)$.

For the forward direction, we need to show that if a parameterized multi-criteria optimization problem Π is FPT, then it admits a kernel. Suppose there is an algorithm that solves instances of Π of size n with parameter k in time $f(k)n^c$, where c is a constant. On input $(I, k, t_1, \ldots, t_r)$, the reduction algorithm runs the FPT algorithm for n^{c+1} steps. If the reduction algorithm terminates after at most n^{c+1} steps, the reduction algorithm outputs an instance $(I', k', t'_1, \ldots, t'_r)$ of constant size which is hard-coded in the reduction algorithm and does not depend on the input instance $(I, k, t_1, \ldots, t_r)$. Thus $|I'| + k' + t'_1 + \cdots + t'_r$ is upper bounded by a constant. If the FPT algorithm does not terminate after n^{c+1} steps, the reduction algorithm halts and outputs the instance $(I, k, t_1, \ldots t_r)$. This is because, in this case, $f(k)n^c \geq n^{c+1}$ implying $n \leq f(k)$.

We now describe the solution-lifting algorithm. If the reduction algorithm returns $(I, k, t_1, \ldots t_r)$ the solution-lifting algorithm outputs the solution it receives as input. If the reduction algorithm outputs $(I', k', t'_1, \ldots, t'_r)$, then it implies the FPT algorithm terminated in polynomial time, which means that the solution-lifting algorithm can use the FPT algorithm to output an optimal solution to $(I, k, t_1, \ldots t_r)$ in polynomial time, regardless of the the solution to $(I', k', t'_1, \ldots, t'_r)$ it gets as input. This concludes the proof. □

We now define a parameterized approximation algorithm for parameterized multi-criteria optimization problems.

Definition 20 (Parameterized $(\alpha_0, \ldots, \alpha_r)$-approximation algorithm for a parameterized multi-criteria optimization problem). *Let $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ be a tuple where each $\alpha_i \geq 1$ is a constant. A* fixed parameter tractable $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximation algorithm *for a parameterized multi-criteria optimization problem Π is an algorithm that takes as input an instance $(I, k, t_1, \ldots t_r)$, runs in time $f(k)(|I| + t_1 + \cdots + t_r)^{\mathcal{O}(1)}$, and outputs an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate solution s. A string s is said to be an* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate solution *if the following conditions are satisfied*

- *If Π is a minimization problem*

1. $w(I,k,s) \leq \alpha_0 \mathsf{OPT}(I,k,t_1,\ldots t_r)$
2. $\forall i \in [r]$, $\ell^i(I,k,s) \geq t_i/\alpha_i$ *or* $\ell^i(I,k,s) \leq \alpha_i t_i$

- *If* Π *is a maximization problem*

1. $w(I,k,s) \geq \mathsf{OPT}(I,k,t_1,\ldots t_r)/\alpha$
2. $\forall i \in [r]$, $\ell^i(I,k,s) \geq t_i/\alpha_i$ *or* $\ell^i(I,k,s) \leq \alpha_i t_i$

Remark 4. *Note that the definition can be extended to tuples* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ *where each* α_i, $i \in [r]$, *can depend on the parameter* k, *on the size of the instance* $|I| + \sum_{i\in[r]} t_i$, *or both.*

Definition 21 (**$(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate polynomial time preprocessing algorithm ($(\alpha_0, \alpha_1, \ldots, \alpha_r)$-APPA) for a parameterized multi-criteria optimization problem**). *Let* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ *be a tuple where each* $\alpha_i \geq 1$ *is a constant. An* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate polynomial-time preprocessing algorithm $\mathcal{A}$ *for* Π *is a pair of polynomial time algorithms. The first one is called the* reduction algorithm *and it computes a map* $\mathcal{R}_{\mathcal{A}} \colon \Sigma^\star \times \mathbb{N}^{r+1} \to \Sigma^\star \times \mathbb{N}^{r+1}$. *Given as input an instance* $(I,k,t_1,\ldots,t_r)$ *of* Π, *the reduction algorithm outputs another instance* $(I',k',t'_1,\ldots,t'_r) = \mathcal{R}_{\mathcal{A}}(I,k,t_1,\ldots,t_r)$.

The second algorithm is called the solution-lifting algorithm. *This algorithm takes as input an instance* $(I,k,t_1,\ldots,t_r) \in \Sigma^\star \times \mathbb{N}^{r+1}$ *of* Π, *the output instance* $(I',k',t'_1,\ldots,t'_r)$ *of the reduction algorithm and a solution* s' *to the instance* $(I',k',t'_1,\ldots,t'_r)$. *Suppose* s' *is a* $(\beta_0,\beta_1,\ldots,\beta_r)$*-approximate solution. Then the solution-lifting algorithm runs in time polynomial in the size of its input and outputs a* $(\alpha_0\beta_0, \alpha_1\beta_1, \ldots, \alpha_r\beta_r)$*-approximate solution* s *to* $(I,k,t_1,\ldots,t_r)$.

Definition 22 (**$(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate kernelization for a parameterized multi-criteria optimization problem**). *An* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate kernelization (or an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-kernel) *for a parameterized multi-criteria optimization problem* Π *and a tuple* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ *with* $\alpha_i \geq 1$ *for all* $i \in [0,r]$ *is an* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-APPA* $\mathcal{A}$ *such that* $size_{\mathcal{A}}$ *is upper bounded by a computable function* $g : \mathbb{N} \to \mathbb{N}$ *of* k.

When the function g is polynomial it is called an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$- *approximate polynomial kernel.*

Proposition 9. *For every tuple* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ *with* $\alpha_i \geq 1$ *for all* $i \in [0,r]$ *and decidable parameterized multi-criteria optimization problem* Π, Π *admits a fixed parameter tractable* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-approximation algorithm if and only if* Π *has an* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-approximate kernel.*

Proof. First, we prove the backward direction. Given an instance $(I, k, t_1, \ldots, t_r)$, one may run the reduction algorithm to obtain a new instance $(I', k', t'_1, \ldots, t'_r)$ of size bounded by a computable function of k, say g. Since $(I', k', t'_1, \ldots, t'_r)$ has bounded size and Π is decidable, an optimal solution s' can be found in time at most $g'(k)$ for some computable function g'. Finally one can use the solution-lifting algorithm to obtain an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate solution s to $(I, k, t_1, \ldots, t_r)$.

For the forward direction, we need to show that if a parameterized multi-criteria optimization problem Π admits a fixed parameter tractable $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximation algorithm then it admits an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate kernel. Suppose there is an algorithm that given instances of Π of size n with parameter k runs in time $f(k)n^c$, where c is a constant, and outputs an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate solution. On input $(I, k, t_1, \ldots, t_r)$, the reduction algorithm runs the FPT $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximation algorithm for n^{c+1} steps. If the reduction algorithm terminates after at most n^{c+1} steps, the reduction algorithm outputs an instance $(I', k', t'_1, \ldots, t'_r)$ of constant size which is hard-coded in the reduction algorithm and does not depend on the input instance $(I, k, t_1, \ldots, t_r)$. Thus $|I'| + k' + t'_1 + \cdots + t'_r$ is upper bounded by a constant. If the FPT approximation algorithm does not terminate after n^{c+1} steps the reduction algorithm halts and outputs the instance $(I, k, t_1, \ldots t_r)$. This is because, in this case, $f(k)n^c \geq n^{c+1}$ implying $n \leq f(k)$.

We now describe the solution-lifting algorithm. If the reduction algorithm returns $(I, k, t_1, \ldots t_r)$, the solution-lifting algorithm outputs the solution it receives as input. If the reduction algorithm outputs $(I', k', t'_1, \ldots, t'_r)$, then it implies the FPT $(\alpha_0, \ldots \alpha_r)$-approximation algorithm terminated in polynomial time, which means that the solution-lifting algorithm can use the FPT $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximation algorithm to output an $(\alpha_0, \ldots \alpha_r)$-approximate solution to $(I, k, t_1, \ldots t_r)$ in polynomial time, regardless of the the solution to $(I', k', t'_1, \ldots, t'_r)$ it gets as input. This concludes the proof. □

This also implies the following theorem.

Theorem 18. *For every tuple $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ with $\alpha_i \geq 1$ for all $i \in [0, r]$ and decidable parameterized multi-criteria optimization problem Π, Π admits a polynomial time $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximation algorithm if and only if Π has an $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-approximate kernel of constant size.*

Definition 23 (Strict $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-APPA for a parameterized multi-criteria optimization problem). *Let $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ be a tuple where each $\alpha_i \geq 1$*

is a constant. An $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-APPA* $\mathcal{A}$ *for* Π *is said to be* strict *if for every instance* $(I, k, t_1, \ldots, t_r)$*, reduced instance* $(I', k', t'_1, \ldots, t'_r) = \mathcal{R}_{\mathcal{A}}(I, k, t_1, \ldots, t_r)$ *and* $(\beta_0, \beta_1, \ldots, \beta_r)$*-approximate solution* s' *to* $(I', k', t'_1, \ldots, t'_r)$*, for any* $\beta_i \geq 1$*,* $i \in [0, r]$*, the string* s *returned by the solution-lifting algorithm, when given* s' *as input, is*

$$(\max\{\alpha_0, \beta_0\}, \max\{\alpha_1, \beta_1\}, \ldots, \max\{\alpha_r, \beta_r\})\text{-approximate solution}$$

A reduction rule for a parameterized multi-criteria optimization problem is a polynomial time algorithm that computes a map $\mathcal{R}_{\mathcal{A}} \colon \Sigma^\star \times \mathbb{N}^{r+1} \to \Sigma^\star \times \mathbb{N}^{r+1}$.

Definition 24. *A reduction rule is said to be* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-safe *for* Π *if there exists a solution-lifting algorithm, such that the rule, together with the solution-lifting algorithm, constitutes a strict* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-APPA for* Π*. A reduction rule is said to be simply* safe *if it is* $(1, \ldots, 1)$*-safe.*

Definition 25. *An* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-approximate kernel* $\mathcal{A}$ *is called* strict *if* $\mathcal{A}$ *is a strict* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-APPA.*

Note that strict $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-APPAs are closed under composition; that is, the composition of two strict $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-APPAs is again a strict $(\alpha_0, \alpha_1, \ldots, \alpha_r)$-APPA.

Definition 26 (Polynomial size approximate kernelization scheme (PSAKS) for a parameterized multi-criteria optimization problem). *A polynomial size approximate kernelization scheme (PSAKS) for a parameterized multi-criteria optimization problem* Π *is a family of* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-approximate kernelization algorithms, with one such algorithm for each tuple* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*, where each* $\alpha_i > 1$*.*

Remark 5. *PSAKS can be uniform or non-uniform. A uniform PSAKS can be thought of as a single* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$*-approximate kernelization algorithm, where the tuple* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ *is a part of the input. For an input* $(I, k, t_1, \ldots, t_r)$*, the size of the output instances of a PSAKS is upper bounded by* $f(\alpha_0, \ldots, \alpha_r)k^{g(\alpha_0, \ldots, \alpha_r)}$*, where* f *and* g *are computable functions that are independent of* $|I|, k$ *or* t_i*s,* $i \in [r]$*.*

Definition 27 (Efficient Polynomial size approximate kernelization scheme (EPSAKS) for a parameterized multi-criteria optimization problem). *A size efficient or just an efficient PSAKS (EPSAKS) for a parameterized multi-criteria optimization problem* Π *is a PSAKS where the size of the output instances of the reduction algorithm on an input* $(I, k, t_1, \ldots, t_r)$ *and a tuple* $(\alpha_0, \ldots, \alpha_r)$ *is upper bounded by* $f(\alpha_0, \ldots, \alpha_r)k^c$*. Here, the computable function* f *is independent*

of $|I|, k$ *or* $t_i s$, $i \in [r]$ *and the constant* c *is independent of* $|I|, k$, $t_i s$, $i \in [r]$ *or* $\alpha_j s$, $j \in [0, r]$.

Chapter 12

Lossy kernels for Constrained Satisfiability

In the final technical chapter of this thesis, we use the multi-criteria lossy kernelization framework developed in the previous chapter to design polynomial lossy kernels for variants of the CC-MaxSat problem under budget, fairness and matroid constraints. We begin by presenting a high-level sketch of our algorithm for designing a lossy kernel for F-Max k-weight SAT parameterized by the solution size, and later describe a simple modification that extends this approach to handle the additional matroid constraint. In Chapter 12.2, we then design an EPSAKS for the problem of F-Max k-weight SAT parameterized by the solution size. Finally, in Chapter 12.3, we present an EPSAKS for (M, F)-Max k-weight SAT when parameterized by the solution size.

12.1 An overview of the lossy kernel

We present a rough sketch of our algorithm for designing a lossy kernel for the F-Max k-weight SAT problem parameterized by k. We later describe a simple modification that extends the approach to handle the additional matroid constraint. Our algorithm proceeds in three main steps:

Phase I: Reducing the number of negative variables

In the first step, we aim to bound the number of variables that appear as negative literals in the formula. We call such variables as negative variables and denote them by the set $\mathcal{V}^{neg}$. To do this, we use the notion of *normalized negative degree*, introduced by [58], which allows us to quantify the relative effectiveness of a variable in satisfying clauses as a negative literal. We make the following key observation: if for a given color class $j \in [r]$, there are sufficiently many (at least $\frac{2k^2}{\varepsilon} + k$) variables, denoted by Y^j_{high}, with high normalized negative degree (at least $\frac{\varepsilon}{2k} t_j$), then any assignment of weight at most k together, satisfy the required quota of t_j clauses.

As a result, we can safely ignore such color classes during further processing as their quota t_js are satisfied automatically by any assignment of weight at most k.

Henceforth, we may assume that for any color class j, the number of negative variables with high normalized negative degree are bounded (at most $\frac{2k^2}{\varepsilon} + k$). Since there are only r color classes, the total number of negative variables with high normalized negative degree across all color classes is at most $\mathcal{O}(\frac{rk^2}{\varepsilon})$. And the remaining variables—those with low normalized negative degree in all color classes— each has only limited ability to help satisfy clauses as negative literals in any of these color classes.

Now, consider any arbitrary assignment of weight at most k. This assignment sets all but at most k of these remaining variables to 1. Let $\mathcal{C}_{\text{del}}$ be the set of clauses where any of these variables are present as a negative literal. Hence, the number of clauses in $\mathcal{C}_{\text{del},j}$, the subset of $\mathcal{C}_{\text{del}}$ restricted to clauses of color class j, that might be unsatisfied by the assignment is proportionally small compared to t_js.

Thus, even if we assume that all the clauses in $\mathcal{C}_{\text{del}}$ are satisfied, the additive error introduced is small and can be absorbed in the approximation guarantee. However, we cannot simply delete these variables, since many of them may have high positive degree—i.e., they appear positively in many clauses, and could significantly contribute to satisfying those color class requirements t_js. Hence, we do not delete the variables; instead, we only remove the clauses ($\mathcal{C}_{\text{del}}$) where they appear negatively, thereby killing off their small influence as possible negative literals while preserving their potential as positive literals. As a result of this step, the number of variables that appear anywhere in the formula as negative literals is now bounded by $\mathcal{O}(\frac{rk^2}{\varepsilon})$.

Phase II: Reducing the number of positive variables

The next step of our algorithm deals with bounding the number of positive variables in the formula. Recall that a positive variable occurs only as a positive literal in any clause in the formula. We denote the set by $\mathcal{V}^{pos}$. We handle color constraints by a technique called the *bucketing procedure* which aggregates positive variables based on the frequency of their occurrences in clauses of every color. The procedure, thus, partitions the set of positive variables into equivalence classes or buckets, where each bucket contains variables that occur in similar number of clauses, for every color. We incorporate a finer implementation of the bucketing procedure by first partitioning the set of colors into two categories based on their satisfiability requirements. If the satisfiability requirement for a color is high (when $t_j \geq \frac{4kd}{\varepsilon^2}$), then we categorize the positive variables into buckets where each bucket contains positive variables that appear in "almost" the same number of clauses of every color. However, if the satisfiability requirement for a color is low (when $t_j < \frac{4kd}{\varepsilon^2}$), then we categorize the positive variables into buckets where each bucket contains positive variables that appear in exactly the same number of clauses of every color. A clever implementation of the procedure leads to the number of buckets being polynomial in k, the input instance parameter and ε, the error parameter ($\mathcal{O}\left(\frac{kd}{\varepsilon^2}\right)^r$). The procedure accommodates us to focus on a single bucket at a time.

Now, to bound the size of every bucket, we crucially use two important concepts: (a) the sunflower lemma to deal with low satisfiability requirement color classes [58], and (b) the fact that the number of variables that occur together in a lot of clauses with a particular variable (at least $\frac{\varepsilon t_j}{k+|\mathcal{V}^{neg}|}$ clauses) is bounded when the formula admits a sparse structure (for eg. Lemma 6 gives such a bound when the formula is $K_{d,d}$-free). By the structure of a formula, we mean its clause variable incidence graph. If the incidence graph is sparse, the corresponding formula is said to admit a sparse structure. If the size of a bucket is more than a set threshold (q_{th}), then we delete a variable from the bucket. We discuss how this threshold is set in the subsequent paragraphs. In case the variable deleted belongs to a hypothetical optimum solution, we show the existence of enough variables in the bucket to replace the deleted variable such that the loss incurred upon replacement is bounded. The invariant we maintain is that after every deletion of a variable from the bucket, there exist k variables that are "good" candidates for the replaced/deleted variable. The reason we maintain k such variables at every point is to consider the case when all the variables that are set to 1 by some hypothetical optimum assignment belong to the same bucket.

When a bucket contains variables that occur in at least one clause of any low satisfiability requirement color class, we make use of the *sunflower rule*. If the bucket contains "sufficiently high" number of variables (at least $q_{\text{th}} := d((q_{\text{sf}}-1)r\frac{4kd}{\varepsilon^2})^d)$, then we can find a combinatorial structure called a sunflower of a certain size (q_{sf}) in the bucket. Loosely speaking, the set system underlying the sunflower uses as its ground set the union of clauses from low-requirement color classes. The associated family contains, for each variable in the bucket, a set consisting of all low-requirement color clauses that include that variable. Since satisfiability requirement is low for such colors, the number of clauses of such type that can be satisfied by an optimal assignment of weight at most k is also bounded (at most $(k+|\mathcal{V}^{\text{neg}}|)r \cdot \frac{4kd}{\varepsilon^2}$). Thus, keeping a sunflower with enough petals, i.e., of a large enough size, guarantees the existence of k replacement variables. Now to deal with high satisfiability requirement color classes in this case, we make use of the concept of *heavy neighbors* of a variable. For a variable, its heavy neighbors are those variables that occur in a large fraction of the clauses together with the mentioned variable. Since the formulae we are working on admit a sparse structure, the number of heavy neighbors are bounded.

Thus, the number of heavy neighbors of the variables that are either set to 1 by an optimum assignment or occur as negative variables is also bounded (at most $(k+|\mathcal{V}^{\text{neg}}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{\text{neg}}|)}{\varepsilon}\right)^d$). Thus, increasing the size of the sunflower to also incorporate the bounded number of heavy neighbors discussed above guarantees the existence of k replacement variables. Hence, we set $q_{\text{sf}} = (k+|\mathcal{V}^{\text{neg}}|)r \cdot \frac{4kd}{\varepsilon^2} + (k+|\mathcal{V}^{\text{neg}}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{\text{neg}}|)}{\varepsilon}\right)^d + k + 1$.

On the other hand, if the bucket contains variables that do not occur in any clause of low satisfiability requirement color classes, then, only the heavy neighborhood bound is sufficient for us. Since our threshold (q_{th}) is much larger than the heavy neighbors bound ($(k+|\mathcal{V}^{\text{neg}}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{\text{neg}}|)}{\varepsilon}\right)^d$), there exist k replacement variables in this case as well. We want to remark that we do not use a sunflower to select a variable to delete in this case. We arbitrarily pick a variable to delete if the size of the bucket exceeds the said threshold.

Note that this procedure preserves the set of negative variables in the formula. The set of positive variables in the formula after this procedure is bounded by $\mathcal{O}\left(\frac{k(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}$.

Phase III: Reducing the number of clauses

In the final step, we provide an algorithm to bound the set of clauses by a function of the number of variables in the formula. We use a well-known technique called *scaling and rounding of weights* when we have a multiset of clauses. We appropriately set a scaling factor to scale down the duplicates of each distinct clause. We would like to remark we have a set of clauses, in contrast to a multiset, then the set of clauses is already bounded by a function of the number of variables due to the sparseness of the formula. The set of clauses gets bounded by $|\mathcal{V}|^{\mathcal{O}(d)}$ after this procedure.

Phase IV: Combining everything

We now combine the above steps to get an approximate kernel. The first algorithm returns a formula where the number of negative variables is $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$. The second algorithm bounds the number of positive variables by $\left(\frac{k(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2r)} = \left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^2r)}$. Finally, the third algorithm bounds the number of clauses by $|\mathcal{V}|^{\mathcal{O}(d)} = \left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^3r)}$. Thus, the reduced instance has $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^2r)}$ variables and $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^3r)}$ clauses.

Handling the matroid constraint

We next present the modifications needed to our algorithm for F-Max k-weight SAT to design a lossy kernel for the (M, F)-Max k-weight SAT problem.

With the addition of a matroid constraint, the solution set S must now be independent in the given matroid $\mathcal{M} = (\mathcal{V}, I)$, which is defined on the set of variables. We assume oracle access to a matroid of rank k. We also assume each variable in $\mathcal{V}$ to be an independent set as a singleton. Otherwise, we can set those variables to 0 and appropriately remove them clauses that contain them as positive literals, and remove the clauses containing them as negative literals as such clauses will be satisfied. In the case of only partition constraints, it was sufficient to replace a deleted variable with another variable of "similar" satisfying ability. However, in the presence of a matroid constraint, such naive replacement is no longer sufficient. To handle this, we rely on the well-known representative family result for matroids.

Observe that the procedures bounding the set of negative variables and clauses of our algorithm (for the partition only constraints) remain unaffected by the matroid constraint, as these steps involve only clause deletion—not variable deletion. Thus,

the entire set of variables remains available for constructing a solution that satisfies the independence condition. The key modification occurs when we delete positive variables. In the case without matroid constraints (in the partition only setting), we applied a reduction based on bucketing followed by identifying a sunflower of size q_{sf} and removing an arbitrary variable from the sunflower. However, with the matroid constraint, deleting an arbitrary variable may be problematic as even though we will be able to find a replacement positive variable with similar satisfying ability, such a variable may not be independent with the remainder of the solution. To handle this issue, we compute a larger sunflower (of size $k \cdot q_{\text{sf}} + 1$) and proceed a bit more carefully. Note that if a bucket contains "sufficiently high" (higher than the partition only setting) number of variables (at least $q_{\text{th}} \coloneqq d((kq_{\text{sf}})r\frac{4kd}{\varepsilon^2})^d)$, then we can find the required larger sunflower of size $(k \cdot q_{\text{sf}} + 1)$ in the bucket. And, although the overall size of the required kernel increases due to the larger buckets, it still remains polynomial in the relevant parameters.

We now describe the reduction procedure used to bound the number of positive variables following the identification of such a larger sunflower $\mathcal{P}$. We begin by repeatedly computing disjoint $(k-1)$-representative families of variables involved in this sunflower as follows. First, we compute a $(k-1)$-representative family, Q_1 of $\mathcal{P}$. Then, we compute the $(k-1)$-representative families Q_2 of $\mathcal{P} \setminus Q_1$, Q_3 of $\mathcal{P} \setminus (Q_1 \cup Q_2)$ and so on until we compute q_{sf} many $(k-1)$- representative families in the process, each of size at most k. We delete an arbitrary variable $y \notin Q_1 \uplus Q_2 \ldots \uplus Q_{q_{\text{sf}}}$ from $\mathcal{P}$. Now, following the arguments from the partition-only setting, for the deleted variable y, there exists a representative family Q_i, where $i \in [q_{\text{sf}}]$, from which every variable is a potential replacement in a desired approximate solution (in terms of satisfying ability)—ignoring the matroid constraint. However, since Q_i is a representative family (of $\{y\}$ as well), we are guaranteed that at least one of these replacements not only maintains approximate satisfying requirements for all the color classes, but also ensures that the resulting set (replacement variable along with the remainder of the solution) remains independent in the matroid. Hence, we are able to maintain both the satisfiability requirements and the matroid independence constraint simultaneously.

12.2 Lossy kernel for F-Max k-weight SAT

Next, we design an EPSAKS for the multi-criteria optimization problem F-Max k-weight SAT parameterized by the solution size k. Let $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$

be an instance of F-MAX k-WEIGHT SAT. We formally define the problem below and then formulate it as a multi-criteria optimization problem.

F-MAX k-WEIGHT SAT
Input: An instance $\mathcal{I} = ((\Phi, r, f), k, t)$, where $\Phi = (\mathcal{V}, \mathcal{C})$ is a CNF-SAT formula, $f : \mathcal{C} \to [r]$ is a surjective function, and for each $j \in [r]$, we say that $\mathcal{C}_j \coloneqq \{\mathfrak{s} \in \mathcal{C} : f(\mathfrak{s}) = j\}$ is the set of clauses of *color* j and, $t : [r] \to \mathbb{N}$ is a function such that $t_j \coloneqq t(j) \geq 0$ is the *satisfiability requirement* for *color* j. k is a non-negative integer.
Parameter: k
Question: Does there exist an assignment β of weight at most k to $\mathcal{V}$, such that for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(\beta)| \geq t_j$?
Here, $\mathsf{sat}_{\Phi_j}(\beta) \subseteq \mathcal{C}_j$ is the set of clauses of color $j \in [r]$ that are satisfied by β in the subformula $\Phi_j = (\mathcal{V}, \mathcal{C}_j)$.

Modeling as parameterized multi-criteria minimization problem: Given an instance $\mathcal{I} = ((\Phi, r, f), k, t)$, where $\Phi = (\mathcal{V}, \mathcal{C})$ and $t = (t_1, \ldots, t_r)$, we model the above problem as a parameterized multi-criteria minimization problem as follows. For a given set of variables $S \subseteq \mathcal{V}$,

- the main optimization function w is defined as

$$w((\Phi, r, f), k, S) = \min\{|S|, k+1\},$$

- the r constraint functions are defined as: for $j \in [r]$,

$$\ell^j((\Phi, r, f), k, S) = \begin{cases} -\infty & |S| > k \\ |\mathsf{sat}_{\Phi_j}(S)| & \text{otherwise} \end{cases}$$

S is a solution to $((\Phi, r, f), k, t)$ if, for all $j \in [r]$, $\ell^j((\Phi, r, f), k, S) \geq t_j$.

In this paper, our goal is to design an EPSAKS for this problem. More specifically, we give a $(1, 1+\varepsilon, \ldots 1+\varepsilon)$-approximate kernel, for every $\varepsilon > 0$, for F-MAX k-WEIGHT SAT when the input formula Φ is $K_{d,d}$-free.

A brief overview of the approximate kernelization procedure: We design our approximate kernel in three main steps. The first step is to design an APPA

that bounds the number of negative variables present in the instance returned by the reduction algorithm by a polynomial function of k and ε. In the next step, we design an APPA that bounds the number of positive variables as a polynomial function of k, ε, and preserves the set of negative variables. The third APPA bounds the number of clauses by a polynomial function of k, ε, and the number of variables. We describe each step in the following three subsections and then prove the following theorem.

Theorem 19. *For any $\varepsilon \in (0,1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-approximate polynomial kernel for* F-MAX k-WEIGHT SAT*, with $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(rd^2)}$ variables and $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(rd^3)}$ clauses, when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free.*

12.2.1 Bounding the number of negative variables.

We begin by giving a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA, for any $\varepsilon > 0$, for F-MAX k-WEIGHT SAT. The size of the reduced instance given by the APPA has bounded number of negative variables.

Lemma 20. *For any $\varepsilon \in (0,1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA consisting of the pair $(\mathcal{R}_{neg}, \mathsf{Iden})$ for* F-MAX k-WEIGHT SAT*, when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free. The reduction algorithm takes as input an instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ and outputs an instance $\mathcal{I}' = ((\Phi', r, f), k, t')$ where $\Phi' = (\mathcal{V}', \mathcal{C}')$ is a $K_{d,d}$-free formula with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{C}' \subseteq \mathcal{C}$ and the number of negative variables in $\mathcal{V}'$ is at most $\mathcal{O}(\frac{rk^2}{\varepsilon})$.*

First, we provide a brief outline of the APPA as follows:

Reduction Algorithm: The reduction algorithm consists of Reduction Rule 10 which deletes any clause that contains at least $k+1$ negative literals, since these clauses will be satisfied by any assignment of weight at most k. This step is followed by a greedy procedure described in Algorithm 12 that returns a formula with $\mathcal{O}(\frac{rk^2}{\varepsilon})$ negative variables. The intuition behind the greedy procedure is that if the number of variables with high negative degree with respect to a color j is sufficient, then any assignment of weight at most k satisfies the required number of clauses of color j. On the other hand, any assignment of weight at most k to the variables of low negative degree does not satisfy some bounded number of clauses. Here, by negative degree, we refer to the normalized negative degree, which is explained formally later.

Solution Lifting Algorithm: The solution lifting algorithm, lden returns the approximate solution of the reduced instance it receives as input.

Given an instance $\mathcal{I} = ((\Phi, r, f), k, t)$ of F-MAX k-WEIGHT SAT, we first apply the following reduction rule on the set of clauses.

Reduction Rule 10. *Given an instance $\mathcal{I} = ((\Phi, r, f), k, t)$ of* F-MAX k-WEIGHT SAT *if there exists a clause $c \in \mathcal{C}$ such that $|\mathsf{neg}(c)| \geq k+1$, then remove c from $\mathcal{C}$. Let $f(c) = j$, the update t_j to $t_j - 1$ and return the instance $\mathcal{I} = ((\Phi', r, f), k, t')$, where $\Phi' = (\mathcal{V}, \mathcal{C} \setminus \{c\})$.*

Notice that Reduction Rule 10 is safe as any assignment of weight at most k will always assign 0 to at least one of the $k+1$ variables in $\mathsf{neg}(c)$, effectively satisfying it.

Next, we define a concept similar to the one in [58] that captures the contribution of a negative variable towards the satisfaction of clauses containing it as a negative literal.

Definition 28 (normalized negative degree with respect to colors)**.** *Given $Y \subseteq \mathcal{V}$ and $j \in [r]$, the normalized negative degree of a variable $x \in Y$ with respect to the subset Y and color j is defined as follows.*

$$\mathtt{cnndeg}_{Y,j}(x) = \sum_{\substack{c \in \mathcal{C}_j \\ x \in \mathsf{neg}(c)}} \frac{1}{|\mathsf{neg}(c) \cap Y|}$$

We next apply the greedy procedure described in Algorithm 12 and show that it achieves the following. For every color $j \in [r]$, it

- either bounds the set of negative variables with *high* normalized negative degree with respect to that color, or
- shows that any assignment of weight at most k satisfies at least t_j clauses of color class j.

Observation 10. *We infer the following from Algorithm 12.*

- *The set of negative variables with high normalized negative degree with respect to a color j, denoted by Y^j_{high}, is $\mathcal{O}\left(\frac{k^2}{\varepsilon}\right)$.*

Algorithm 12 GREEDY PROCEDURE TO BOUND $\mathcal{V}^{\text{neg}} \subseteq \mathcal{V}(\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t))$

1: Set $Y_{\text{high}} = \emptyset$ and $\mathcal{L} = \emptyset$.
2: **for** $j = 1$ to r **do**
3: Set $\tau_j = \frac{\varepsilon}{2k} t_j$.
4: Let $Y_0^j = \mathcal{V}$ and $i = 1$.
5: **while** there exists an $x_i \in Y_{i-1}^j$ such that $\texttt{cnndeg}_{Y_{i-1}^j, j}(x_i) > \tau_j$ **do**
6: Set $Y_i^j = Y_{i-1}^j \setminus \{x_i\}$ and $i = i + 1$.
7: **end while**
8: Set $Y_{\text{high}}^j = \mathcal{V} \setminus Y_i^j$ and $Y_{\text{low}}^j = Y_i^j$.
9: **if** $|Y_{\text{high}}^j| \leq \frac{2k^2}{\varepsilon} + k$ **then**
10: $Y_{\text{high}} = Y_{\text{high}} \cup Y_{\text{high}}^j$.
11: **else**
12: $\mathcal{L} = \mathcal{L} \cup \{j\}$. ▷ $\mathcal{L}$ stores the set of colors with a large set of high normalized negative degree variables
13: **end if**
14: **end for**
15: Set $Y_{\text{low}} = \bigcap_{j \in [r] \setminus \mathcal{L}} Y_{\text{low}}^j$.
16: Let $\mathcal{C}_{\text{del}} = \{c \in \mathcal{C} \mid \texttt{neg}(c) \cap Y_{\text{low}} \neq \emptyset\} \bigcup (\cup_{j \in \mathcal{L}} \mathcal{C}_j)$ and $\mathcal{C}' = \mathcal{C} \setminus \mathcal{C}_{\text{del}}$.
17: Let $f' = f \restriction_{\mathcal{C}'}$, i.e., f restricted to clauses in $\mathcal{C}'$.
18: Set

$$t'_j = \begin{cases} 0 & \text{if } j \in \mathcal{L} \\ t_j - |\mathcal{C}_{\text{del},j}| & \text{otherwise} \end{cases}$$

Here, $\mathcal{C}_{\text{del},j}$ is the subset of clauses from $\mathcal{C}_j$ that is deleted in Step 16.
19: **return** $((\Phi' = (\mathcal{V}, \mathcal{C}'), r, f'), k, t')$.

- *The set of negative variables with high normalized negative degree with respect to at least one color $j \in [r] \setminus \mathcal{L}$, denoted by Y_{high}, contains at most $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$ variables.*

Claim 6. *For any variable $x \in Y_{high}^j$, $\texttt{cnndeg}_{Y_0^j, j}(x) \geq \frac{\tau_j}{k}$.*

Proof. Let $x_i \in \mathcal{V}$ be the variable that is selected in Step 5 of Algorithm 12 during the i^{th} iteration of the while loop with respect to color $j \in [r]$. Then, its normalized negative degree with respect to the subset Y_{i-1}^j and color j is at least τ_j, i.e., $\texttt{cnndeg}_{Y_{i-1}^j, j}(x_i) > \tau_j$.

Due to Reduction Rule 10, every clause in the formula contains no more than k negative literals. Hence, $\frac{1}{|\texttt{neg}(c) \cap Y|} \in [\frac{1}{k}, 1]$ for every clause c that contains x_i as a negative literal. Moreover, the normalized negative degree of a variable x_i is non-decreasing as the while loop progresses as a negative variable is removed from the

subset Y_ℓ^j during the ℓ^{th} iteration of the while loop, where $\ell \in [i]$. Thus, if the normalized negative degree of a variable x_i with respect to color j at the beginning of the greedy procedure, i.e., with respect to subset Y_0^j, is a_0, then its normalized negative degree with respect to color j at the time of being chosen, i.e., with respect to subset Y_{i-1}^j, is at most $a_0 \cdot k$. This implies that $a_0 \geq \frac{\tau_j}{k}$. □

Claim 7. *If, for some $j \in [r]$, $|Y_{high}^j| > \frac{2k^2}{\varepsilon} + k$, then any assignment of weight at most k to the variables satisfies at least t_j clauses in $\mathcal{C}_j$.*

Proof. Any assignment of weight at most k to $\mathcal{V}$ can set at most k variables to 1. Thus, at least $\frac{2k^2}{\varepsilon}$ variables in Y_{high}^j get assigned 0. We now show that any $\frac{2k^2}{\varepsilon}$ variables from Y_{high}^j satisfy at least t_j clauses in $\mathcal{C}_j$ when they are set to 0. By Claim 6, we know that for a variable $x \in Y_{\text{high}}^j$, its normalized negative degree at the beginning of the greedy procedure with respect to color j is at least $\frac{\tau_j}{k}$. Let $\widetilde{Y}$ be a subset of any $\frac{2k^2}{\varepsilon}$ variables from Y_{high}^j.

$$\sum_{x \in \widetilde{Y}} \mathtt{cnndeg}_{Y_0^j, j}(x) \geq \frac{2k^2}{\varepsilon} \cdot \frac{\tau_j}{k} = t_j$$

Therefore, the number of clauses satisfied by variables in $\widetilde{Y}$ when they are set to 0 are

$$\begin{aligned}
|\bigcup_{x \in \widetilde{Y}} N^{-j}(x)| &= \sum_{x \in \widetilde{Y}} \sum_{\substack{c \in \mathcal{C}_j \\ x \in \mathsf{neg}(c)}} \frac{1}{|\mathsf{neg}(c) \cap \widetilde{Y}|} \\
&\overset{(\text{☺})}{\geq} \sum_{x \in \widetilde{Y}} \sum_{\substack{c \in \mathcal{C}_j \\ x \in \mathsf{neg}(c)}} \frac{1}{|\mathsf{neg}(c) \cap Y_0^j|} \\
&= \sum_{x \in \widetilde{Y}} \mathtt{cnndeg}_{Y_0^j, j}(x) \\
&\geq t_j
\end{aligned}$$

(☺) holds due to the fact that $\widetilde{Y} \subseteq Y_0^j = \mathcal{V}$. □

Lemma 21. *Let $\mathcal{C}_{del}$ be the set of clauses in Step 16 of Algorithm 12 that is deleted and let β be any assignment of weight at most k to $\mathcal{V}$. Then, for each $j \in [r] \setminus \mathcal{L}$, the set of clauses among $\mathcal{C}_{del,j}$ that is unsatisfied by β is at most $\frac{\varepsilon}{2} \cdot t_j$. Here, $\mathcal{C}_{del,j}$ is the subset of $\mathcal{C}_{del}$ restricted to clauses of color class j.*

Proof. Consider the color $j \in [r] \setminus \mathcal{L}$.

Let $\mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)}$ be the set of clauses among $\mathcal{C}_{\text{del},j}$, the clauses deleted by the algorithm from $\mathcal{C}_j$, that is unsatisfied by β. Notice that any clause $c \in \mathcal{C}_{\text{del},j}$ is unsatisfied because $\mathsf{neg}(c) \subseteq \beta^{-1}(1)$. Moreover, any clause c is in $\mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)}$ because $\mathsf{neg}(c) \cap Y_{\text{low}}^j \neq \emptyset$. Let Y_{low}^j be the set of variables that are retained after the while loop ends in Step 7 of Algorithm 12. Then, Y_{low}^j is the set of variables that have low normalized negative degree with respect to itself and color j.

$$
\begin{aligned}
|\mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)}| &= \sum_{c \in \mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)}} 1 \\
&= \sum_{c \in \mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)}} \sum_{x \in (\mathsf{neg}(c) \cap Y_{\text{low}}^j)} \frac{1}{|\mathsf{neg}(c) \cap Y_{\text{low}}^j|} \\
&\leq \sum_{c \in \mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)}} \sum_{x \in (\beta^{-1}(1) \cap Y_{\text{low}}^j)} \frac{1}{|\mathsf{neg}(c) \cap Y_{\text{low}}^j|} \\
&= \sum_{x \in (\beta^{-1}(1) \cap Y_{\text{low}}^j)} \sum_{\substack{c \in \mathcal{C}_{\text{del},j}^{\text{unsat}(\beta)} \\ x \in \mathsf{neg}(c)}} \frac{1}{|\mathsf{neg}(c) \cap Y_{\text{low}}^j|} \\
&= \sum_{x \in (\beta^{-1}(1) \cap Y_{\text{low}}^j)} \mathtt{cnndeg}_{Y_{\text{low}}^j, j}(x) \\
&\leq \sum_{x \in (\beta^{-1}(1) \cap Y_{\text{low}}^j)} \tau_j \\
&\leq k\tau_j = \frac{\varepsilon}{2} t_j \qquad \text{(as } \beta \text{ has weight at most } k\text{)}
\end{aligned}
$$

□

Observation 11. *The negative variables in the formula of the reduced instance are only the variables in Y_{high} since all the clauses with negative occurrences of variables in Y_{low} are deleted. Thus the number of negative variables in Φ' is $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$ as seen in Observation 10.*

Observation 12. *Suppose there exists $S \subseteq \mathcal{V}$ of size k such that $\mathsf{sat}_{\Phi_j}(S) \geq t_j$ for each $j \in [r]$, then from Algorithm 12, it can be easily observed that in the reduced instance $\mathsf{sat}_{\Phi'_j}(S) \geq t'_j$ since $t'_j \leq t_j - |\mathcal{C}_{del,j}|$ as $|\mathcal{C}_{del,j}|$ clauses of color class j are deleted. Thus, if S is a solution of the input instance then it also is a solution for the reduced instance.*

Lemma 22. *Let S_{apx} be a $(\beta_0, \beta_1 \ldots, \beta_r)$-approximate solution of the reduced in-*

stance for any $(\beta_0, \beta_1 \ldots, \beta_r) \in \mathbb{N}^{r+1}$. *Then,* S_{apx} *is a* $(\beta_0, (1+\varepsilon)\beta_1, \ldots, (1+\varepsilon)\beta_r)$*-approximate solution of the input instance.*

Proof. If for $j \in [r]$, $t'_j = 0$, then it corresponds to a color $j \in \mathcal{L}$. Then due to Claim 7, we get $\mathsf{sat}_{\Phi_j}(S_{apx}) \geq t_j$.

Now, in the other case, when $j \in [r] \setminus \mathcal{L}$, we have $\mathsf{sat}_{\Phi'_j}(S_{apx}) \geq \frac{t'_j}{\beta_j}$.

We consider two cases here. When $|\mathcal{C}_{\mathrm{del},j}| \geq \frac{\varepsilon}{2} t_j$.

$$
\begin{aligned}
\mathsf{sat}_{\Phi_j}(S_{apx}) &\geq \frac{t'_j}{\beta_j} + \left(|\mathcal{C}_{\mathrm{del},j}| - \frac{\varepsilon}{2} t_j\right) && \text{(from Lemma 21)} \\
&\geq \frac{t_j - |\mathcal{C}_{\mathrm{del},j}|}{\beta_j} + |\mathcal{C}_{\mathrm{del},j}| - \frac{\varepsilon}{2} t_j \\
&= \frac{t_j}{\beta_j} - \frac{|\mathcal{C}_{\mathrm{del},j}|}{\beta_j} + |\mathcal{C}_{\mathrm{del},j}| - \frac{\varepsilon}{2} t_j \\
&\geq \frac{t_j}{\beta_j} + |\mathcal{C}_{\mathrm{del},j}| \left(1 - \frac{1}{\beta_j}\right) - \frac{\varepsilon}{2} t_j \\
&\geq \frac{t_j}{\beta_j} + \frac{\varepsilon}{2} t_j \left(1 - \frac{1}{\beta_j}\right) - \frac{\varepsilon}{2} t_j && \text{(assuming } |\mathcal{C}_{\mathrm{del},j}| \geq \tfrac{\varepsilon}{2} t_j\text{)} \\
&= \frac{t_j}{\beta_j} - \frac{\varepsilon}{2\beta_j} t_j \\
&= \frac{t_j}{\beta_j} \left(1 - \frac{\varepsilon}{2}\right) \\
&\geq \frac{t_j}{\beta_j (1+\varepsilon)}.
\end{aligned}
$$

When $|\mathcal{C}_{\mathrm{del},j}| \leq \frac{\varepsilon}{2} t_j$ then since a small number of clauses are being deleted in the reduction algorithm, we ignore their contribution while lifting the solution and hence get the following.

$$
\begin{aligned}
\mathsf{sat}_{\Phi_j}(S_{apx}) &\geq \frac{t'_j}{\beta_j} \\
&\geq \frac{t_j - |\mathcal{C}_{\mathrm{del},j}|}{\beta_j} \\
&= \frac{t_j}{\beta_j} - \frac{|\mathcal{C}_{\mathrm{del},j}|}{\beta_j} \\
&\geq \frac{t_j}{\beta_j} - \frac{\frac{\varepsilon}{2}t_j}{\beta_j} && \text{(assuming } |\mathcal{C}_{\mathrm{del},j}| < \tfrac{\varepsilon}{2}t_j\text{)} \\
&= \frac{t_j}{\beta_j}\left(1 - \frac{\varepsilon}{2}\right) \\
&\geq \frac{t_j}{\beta_j(1+\varepsilon)}.
\end{aligned}
$$

Let S'_{opt} be an optimal solution of the reduced instance and S_{opt} be an optimal solution of the input instance. Then we have $|S_{apx}| \leq \beta_0|S'_{opt}| \leq \beta_0|S_{opt}|$. The second inequality follows from the fact that any solution S of the input instance is also a solution of the reduced instance as shown in Observation 12. Thus, we have shown that S_{apx} is a $(\beta_0, (1+\varepsilon)\beta_1, \ldots, (1+\varepsilon)\beta_r)$-approximate solution of the input instance. □

By Lemma 22, we know that any $(\beta_0, \beta_1 \ldots, \beta_r)$-approximate solution S_{apx} to the reduced instance is also a $(\beta_0, (1+\varepsilon)\beta_1, \ldots, (1+\varepsilon)\beta_r)$-approximate solution of the input instance. Thus, along with Observation 11 which states that the number of negative variables in the reduced instance in $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$, Lemma 20 holds.

12.2.2 Bounding the number of positive variables.

In this subsection, we provide a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA for any $\varepsilon > 0$, for an instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ of F-Max k-weight SAT when the input formula Φ is $K_{d,d}$-free. This APPA bounds the size of $\mathcal{V}^{\mathrm{pos}}$ by $f(k, \varepsilon, |\mathcal{V}^{neg}|)$ for some computable polynomial function f. Recall that for any $v \in \mathcal{V}^{pos}$, $d^-(v) = 0$. In particular, we prove the following lemma.

Lemma 23. *For any $\varepsilon \in (0,1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA consisting of the pair $(\mathcal{R}_{pos}, \mathsf{lden})$ for* F-Max k-weight SAT*, when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free. The reduction algorithm takes as input an instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ and outputs an instance $\mathcal{I}' = ((\Phi', r, f), k, t')$ where $\Phi' = (\mathcal{V}', \mathcal{C}')$ is a $K_{d,d}$-free formula with $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{C}' \subseteq \mathcal{C}$, the number of positive variables in $\mathcal{V}'$ is*

at most $\mathcal{O}\left(\frac{k(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}$, *and* $t'_j = \frac{t_j}{(1+\varepsilon)}$ *for all* $j \in [r]$. *It preserves the set of negative variables in* $\mathcal{V}$.

We provide a brief outline of our reduction algorithm and solution lifting algorithm below and then proceed to describe each step formally.

Reduction Algorithm: The reduction algorithm $\mathcal{R}_{pos}$ consists of the following steps.

1) **Truncating Frequency:** We implement a preprocessing step to truncate the frequency of every variable in $\mathcal{V}$. After exhaustive application of this preprocessing step, the frequency of every variable is at most $2t_j$.

2) **Bucketing Procedure:** We first partition the set of colors $[r]$ into two categories, based on their satisfiability requirements (high or low). Then, we implement a procedure to partition $\mathcal{V}^{pos}$ based on its *colored degree*, which we call as *bucketing*. Each partition of $\mathcal{V}^{pos}$ under this procedure is called a *bucket*. The number of buckets created is a polynomial function of k and ε. Variable from $\mathcal{V}^{pos}$ in a bucket are present in *almost* the same number of clauses of any high requirement color j, and in exactly the same number of clauses of any low requirement color j.

3) **Filtering via Sunflower:** For each bucket created in the above step, we delete variables by finding a sufficiently large sunflower. After this procedure, the size of each bucket gets bounded by a polynomial function of k, ε and $|\mathcal{V}^{neg}|$.

Solution lifting algorithm: The solution lifting algorithm Iden simply returns the approximate solution of the reduced instance it receives as input.

Now we introduce a few definitions and results before describing each step formally.

Definition 29. *A sunflower with* k petals *and a* core Y *is a collections of sets* $S_1, \ldots, S_k$ *such that* $S_i \cap S_j = Y$ *for all* $i \neq j$. *The sets* $S_i \setminus Y$ *are called petals. While the core may be empty, it is required that the petals be non-empty.*

For a bipartite graph $G = (A, B)$, the sets S_i are neighborhoods of vertices in A. We say $\mathcal{P} \subseteq A$ is a sunflower with a core $Y \subseteq B$ if for every $u, v \in A$, $N(u) \cap N(v) = Y$. We make use of the following lemma to design our next reduction rule.

Lemma 24 ($K_{a,b}$-free Sunflower Lemma, [58]). *For any $w, \ell \in \mathbb{N}$, any $K_{a,b}$-free bipartite graph $G = (A, B, E)$ such that every vertex in A has degree at most ℓ and $|A| \geq a((w-1)\ell)^b$ has a sunflower of size w. Moreover, such a sunflower can be found in polynomial time.*

Definition 30 (Heavy neighbors). *Let $\Phi = (\mathcal{V}, \mathcal{C})$ be a $K_{d,d}$-free formula where $\mathcal{C} = \uplus_{j \in [r]} \mathcal{C}_j$, $k \in \mathbb{N}$, $t = (t_1, \ldots, t_r)$ and $\varepsilon \in (0, 1)$. A pair of variables $u, v \in \mathcal{V}$ are called* heavy neighbors *of each other with respect to color j if $|N^j(u) \cap N^j(v)| \geq \frac{\varepsilon t_j}{k + |\mathcal{V}^{neg}|}$.*

Lemma 25 (Restatement of Lemma 6). *The number of heavy neighbors of a variable in a $K_{d,d}$-free formula with respect to a color is bounded by $(d-1)\left(\frac{2(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^d$.*

The Reduction Algorithm. We now describe the reduction algorithm $\mathcal{R}_{pos}$ in detail.

1. Truncating Frequency: We implement the following preprocessing step to truncate the frequency of a variable in $\mathcal{V}$. After exhaustive application of this preprocessing step, the frequency of every variable is at most $2t_j$.

Reduction Rule 11. *Given an instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$, we describe our reduction rule based on positive and negative frequencies as follows:*

1. *If there exists a variable v with $d^{-j}(v) > t_j$ for some $j \in [r]$, then delete v from an arbitrarily chosen clause $c \in \mathcal{C}_j$ such that $v \in \mathsf{neg}(c)$.*
2. *If there exists a variable v with $d^{+j}(v) > t_j$ for some $j \in [r]$, then delete v from an arbitrarily chosen clause $c \in \mathcal{C}_j$ such that $v \in \mathsf{pos}(c)$.*

In both cases, the rest of the entries of the instance remain unchanged.

Claim 8. *Reduction Rule 11 is safe.*

Proof. Let $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ be the given input instance. The backward direction holds trivially in both cases.

Consider Case 1. In the forward direction, suppose β is an optimal assignment of weight at most k to the variables. Since we delete v from a clause $c \in \mathcal{C}_j$ that

contains it as a negative literal, its negative degree with respect to j remains at least t_j, i.e., $d^{-j}(v) \geq t_j$, in the reduced instance. Thus, if β assigns 0 to v, it will still satisfy at least t_j clauses in $\mathcal{C}_j$ in the reduced instance. If v is assigned 1 by β, then v is not responsible for the satisfaction of c and can be safely deleted.

We have an analogous argument for Case 2 for the forward direction. If an optimal assignment β to $\mathcal{V}$, of weight at most k, assigns 0 to v, then deleting v from c is safe as v cannot be responsible for the satisfaction of c that contains it as a positive literal. On the other hand, if β assigns 1 to v, then as the positive degree of v is still at least t_j with respect to the color j, i.e., $d^{+j}(v) \geq t_j$, in the reduced instance. Thus, it will still satisfy at least t_j clauses in $\mathcal{C}_j$ in the reduced instance. □

2. The Bucketing Procedure: We partition the set of colors $[r]$ into two categories, based on their satisfiability requirements (high or low).

- Set of colors with high satisfiability requirements, $H \coloneqq \left\{j \in [r] : t_j \geq \frac{4kd}{\varepsilon^2}\right\}$.

  We do *approximate* bucketing on $\mathcal{V}^{pos}$ with respect to the colors in H. Let $\lambda \coloneqq \lceil \log_{(1+\varepsilon)} \frac{2k}{\varepsilon} \rceil$. Fix a color $j \in H$. For each $\alpha \in [\lambda]$, we define

$$A(j, \alpha) \coloneqq \left\{ v \in \mathcal{V}^{\text{pos}} : \frac{t_j}{(1+\varepsilon)^{\alpha}} \leq d^{+j}(v) < \frac{t_j}{(1+\varepsilon)^{\alpha-1}} \right\}.$$

  We also define

$$A(j, 0) \coloneqq \left\{ v \in A : d^{+j}(v) \geq t_j \right\},$$

  and

$$A(j, \lambda+1) \coloneqq \left\{ v \in A : d^{+j}(v) < \frac{t_j}{(1+\varepsilon)^{\lambda}} \right\}.$$

- Set of colors with low satisfiability requirements, $L \coloneqq \left\{j \in [r] : t_j < \frac{4kd}{\varepsilon^2}\right\}$.

  We do *exact* bucketing on $\mathcal{V}^{pos}$ with respect to the colors in L. Fix a color $j \in L$. For each $\alpha \in \{0, 1, \ldots, t_j\}$, let $A(j, \alpha) \coloneqq \{v \in \mathcal{V}^{\text{pos}} : d^{+j}(v) = \alpha\}$.

Let $\mathcal{F}_{\text{b}}$ be the family of all possible functions $\mathsf{bucket} : [r] \to \mathbb{N}$ satisfying the following properties:

- for each $j \in H$, $\mathsf{bucket}(j) \in \{0, 1, 2, \ldots, \lambda+1\}$.
- for each $j \in L$, $\mathsf{bucket}(j) \in \{0, 1, 2, \ldots, t_j\}$.

Then, for each $\mathsf{bucket} \in \mathcal{F}_{\mathrm{b}}$, we define a class $C_{\mathsf{bucket}} \coloneqq \bigcap_{j \in [r]} A(j, \mathsf{bucket}(j))$. Notice that $\mathcal{F}_{\mathrm{b}}$ imposes a partitioning on $\mathcal{V}^{\mathrm{pos}}$.

Observation 13. *For any pair of variables $u, v \in C_{\mathsf{bucket}}$, where C_{bucket} is a non-empty bucket of $\mathcal{V}^{pos}$ corresponding to some $\mathsf{bucket} \in \mathcal{F}_b$, we have the following:*

1. *For all $j \in H$ with $\mathsf{bucket}(j) \in [\lambda]$, $\frac{d^{+j}(v)}{(1+\varepsilon)} \leq d^{+j}(u) \leq (1+\varepsilon) d^{+j}(v)$.*
2. *For all $j \in H$ with $\mathsf{bucket}(j) = 0$, both $d^{+j}(u)$ and $d^{+j}(v)$ are at least t_j.*
3. *For all $j \in H$ with $\mathsf{bucket}(j) = \lambda + 1$, both $d^{+j}(u)$ and $d^{+j}(v)$ are at most $\frac{\varepsilon t_j}{2k}$.*
4. *For all $j \in L$, $d^{+j}(u) = d^{+j}(v)$.*

The total number of buckets returned by the bucketing procedure is

$$(\lambda+2)^{|H|} \cdot \prod_{j \in L} (t_j + 1) \leq \mathcal{O}\left(\left(\left\lceil \log_{(1+\varepsilon)} \frac{2k}{\varepsilon} \right\rceil\right)^r\right) \left(\frac{4kd}{\varepsilon^2}\right)^r = \mathcal{O}\left(\left(\frac{kd}{\varepsilon^2}\right)^r\right).$$

3. Filtering via Sunflower: For each non-empty C_{bucket} of $\mathcal{V}^{\mathrm{pos}}$, if the number of variables in it is more than $q_{\mathrm{th}} \coloneqq d((q_{\mathrm{sf}} - 1) r \frac{4kd}{\varepsilon^2})^d$, where

$$q_{\mathrm{sf}} \coloneqq (k + |\mathcal{V}^{neg}|) r \frac{4kd}{\varepsilon^2} + (k + |\mathcal{V}^{neg}|) r (d-1) \left(\frac{2(k + |\mathcal{V}^{neg}|)}{\varepsilon}\right)^d + k + 1,$$

we apply Reduction Rule 12 described below.

For a given formula $\Phi = (\mathcal{V}, \mathcal{C})$, let $G_{\mathsf{bucket},L} = (A, B)$ be the incidence bipartite graph constructed as follows.

- For every $v \in C_{\mathsf{bucket}}$, add a vertex a_v to A.
- For every clause $c \in \mathcal{C}$ such that $f(c) \in L$, add a vertex b_c to B.
- Add an edge between a_v and b_c iff $c \in N(v)$.
- Delete all isolated vertices.

Since the formula Φ is $K_{d,d}$-free, $G_{\mathsf{bucket},L}$ is also $K_{d,d}$-free as it is the incidence graph of a subformula. According to the construction, for a sunflower $\mathcal{P} \subseteq A$, the sets correspond to neighborhoods of vertices from A participating in $\mathcal{P}$, i.e., the family $\{N(a_v) \mid a_v \in \mathcal{P}\}$. The *core* Y of $\mathcal{P}$ represents the set of vertices in B that are

present in the neighborhoods of each vertex a_v that participates in $\mathcal{P}$, i.e., all the vertices in $\bigcap_{a_v \in \mathcal{P}} N(a_v)$. Similarly, for any vertex $a_v \in \mathcal{P}$, the *petal* corresponding to a_v, denoted by $\mathsf{petal}(a_v)$, is $N(a_v) \setminus Y$.

In the language of CNF-SAT formulae, the sets in the sunflower $\mathcal{P} \subseteq A$ correspond to the family of clauses $\{\mathcal{C}[v] \mid a_v \in \mathcal{P}\}$, where $\mathcal{C}[v] = N^+(v) \bigcap (\cup_{j \in L} C_j)$, i.e. $\mathcal{C}[v]$ is the set of clauses from color classes in L that contain v. Notice that as the *bucketing procedure* is implemented only on $\mathcal{V}^{pos}$, $N(v) = N^+(v)$ for any $v \in \mathcal{V}^{pos}$. The *core* Y of $\mathcal{P}$ represents the set of clauses that contain each variable v corresponding to the vertex $a_v \in A$ that participates in $\mathcal{P}$, i.e., all the clauses in $\bigcap_{a_v \in \mathcal{P}} \mathcal{C}[v]$. Similarly, for any variable $v \in C_{\mathsf{bucket}}$, if $a_v \in \mathcal{P}$, then the *petal* denoted by $\mathsf{petal}(a_v)$ is $\mathcal{C}[v] \setminus Y$. Thus, the sunflower $\mathcal{P}$ collects all variables in C_{bucket} such that for any pair $u, v \in C_{\mathsf{bucket}}$, if $a_u, a_v \in \mathcal{P}$, then $\mathcal{C}[u] \cap \mathcal{C}[v]$ is exactly the same.

Due to exhaustive application of Reduction Rule 11, the positive frequency of every variable is at most t_j when restricted to clauses with color j. Thus, the degree of every vertex in A in the incidence graph is bounded by $\sum_{j \in L} t_j$. Since $j \in L$ we have $t_j \leq \frac{4kd}{\varepsilon^2}$ which implies $\sum_{j \in L} t_j \leq r\frac{4kd}{\varepsilon^2}$. If $|C_{\mathsf{bucket}}| > q_{\mathrm{th}}$, then using Lemma 24 with $a = b = d$, $\ell = r\frac{4kd}{\varepsilon^2}$ and $w = q_{\mathrm{sf}}$, a sunflower $\mathcal{P}$ in $G_{\mathsf{bucket},L}$ of size q_{sf} exists.

Reduction Rule 12 (Sunflower rule)**.** *Given an instance* $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$*, for each* $\mathsf{bucket} \in \mathcal{F}_b$*, if* $|C_{\mathsf{bucket}}| > q_{th}$*, then we apply the reduction rule based on two cases:*

1. *if for some* $v \in C_{\mathsf{bucket}}$*, there exists* $j \in L$ *such that* $d^{+j}(v) > 0$*, then we can find a sunflower* $\mathcal{P}$ *in* $G_{\mathsf{bucket},L} = (A, B)$ *of size* q_{sf}*. Let* a_v *be an arbitrary vertex in the sunflower. Remove the corresponding variable* v *from* C_{bucket}*.*
2. *if for some* $v \in C_{\mathsf{bucket}}$*, and for all* $j \in L$*,* $d^{+j}(v) = 0$*, then remove an arbitrary variable from* C_{bucket}*.*

After exhaustively applying the above procedure, return $((\Phi' = (\mathcal{V}', \mathcal{C}'), r, f), k, t')$ where $\mathcal{V}'$ is the set of variables left, $\mathcal{C}' \subseteq \mathcal{C}$, where $\mathcal{C}'$ is the set of clauses restricted to $\mathcal{V}'$, and set $t'_j = \frac{t_j}{1+\varepsilon}$ for all $j \in [r]$.

Remark 6. *While we do not actively delete clauses, a clause* $c \in \mathcal{C}$ *may inadvertently get deleted if all the variables appearing in it may get deleted during the application of Reduction Rules 11 or 12.*

Remark 7. *Due to the construction of* C_{bucket}*, only variables from* $\mathcal{V}^{pos}$ *are being deleted. Thus, the set of negative variables* $\mathcal{V}^{neg}$ *is preserved by the algorithm.*

The reduction algorithm $\mathcal{R}_{pos}$ partitions $\mathcal{V}^{\text{pos}}$ into $\mathcal{O}\left(\frac{kd}{\varepsilon^2}\right)^r$ buckets and each bucket contains no more than q_{th} variables. Hence, the size of $\mathcal{V}^{\text{pos}}$ in the reduced instance is bounded. We have

$$|\mathcal{V}^{'pos}| \leq \left(\frac{k + |\mathcal{V}^{neg}|}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}. \tag{12.1}$$

The solution lifting algorithm. We describe the solution lifting algorithm lden as follows. It takes as input the instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$, the reduced instance $\mathcal{I}' = ((\Phi' = (\mathcal{V}', \mathcal{C}'), r, f), k, t')$ which is the output of the reduction algorithm $\mathcal{R}_{pos}$, and S' which is a $(\beta_0, \beta_1, \ldots, \beta_r)$-approximate solution to $\mathcal{I}'$, for any $(\beta_0, \ldots, \beta_r) \in \mathbb{N}^{r+1}$, where $\beta_i > 1$ for all $i \in [0, r]$. It simply returns the solution S' it receives as input. In the following discussion, we show that S' is a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon))$-approximate solution to $\mathcal{I}$, thus establishing $(\mathcal{R}_{pos}, \mathsf{lden})$ as a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA.

Lemma 26. *Suppose there exists some $S \subseteq \mathcal{V}$ of size k, such that for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(S)| \geq t'_j$, where $0 \leq t'_j \leq t_j$. Consider a C_{bucket} for some $\mathsf{bucket} \in \mathcal{F}_b$ such that $S \cap C_{\mathsf{bucket}} \neq \emptyset$. Let $x \in (S \cap C_{\mathsf{bucket}})$. Let $\widetilde{C}_{\mathsf{bucket}} \subseteq C_{\mathsf{bucket}}$ be the* final subset *of the bucket that is retained after exhaustive application of the Reduction Rules 11 and 12. Then, at least one of the following holds:*

- *$x \in \widetilde{C}_{\mathsf{bucket}}$, or*
- *there exists a set $R \subseteq \widetilde{C}_{\mathsf{bucket}}$ of size k such that, for each $w \in R$, the following properties hold.*
  1. *For each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S - x + w)| \geq \min\left\{t'_j, |\mathsf{sat}_{\Phi_j}(S)| - 2\varepsilon t_j\right\}$*
  2. *For each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S - x + w)| \geq t'_j$.*

Proof. We refer to each application of the procedure in the sunflower rule (Reduction Rule 12) on C_{bucket} as an *iteration*. Let $C^i_{\mathsf{bucket}} \subseteq C_{\mathsf{bucket}}$ be the subset of C_{bucket} after i iterations, where the total number of iterations is $\ell \geq 0$. Note that if $x \in C^\ell_{\mathsf{bucket}} = \widetilde{C}_{\mathsf{bucket}}$, then the statement of the lemma is obviously true. In the rest of the proof, we consider the case when $x \notin C^\ell_{\mathsf{bucket}}$. Let $S' = S \setminus \{x\}$. We will say that a variable $y \in C_{\mathsf{bucket}}$ is *compatible with* S' if it satisfies the following two properties (analogous to properties (1) and (2) in the statement of the lemma):

(a) for each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S' \cup \{y\})| \geq \min\left\{t'_j, |\mathsf{sat}_{\Phi_j}(S)| - 2\varepsilon t_j\right\}$

(b) for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S' \cup \{y\})| \geq t'_j$.

We first prove the following technical claim.

Claim 9. *Let $y \in C^i_{\mathsf{bucket}}$ be a variable that is compatible with S', and suppose y is deleted from C^i_{bucket} by an application of the procedure in Reduction Rule 12, in order to obtain $C^{i+1}_{\mathsf{bucket}} = C^i_{\mathsf{bucket}} \setminus \{y\}$. Then there exists a subset $R^i \subseteq C^{i+1}_{\mathsf{bucket}}$ of size k, such that each $w \in R^i$ is compatible with S'.*

Proof. **Case A:** Consider a bucket for which there exists at least one color $j \in L$ such that the variables in that bucket appear as positive literals in some clause of color j.

Consider the subgraph $G'_{\mathsf{bucket},L}$ of the graph $G_{\mathsf{bucket},L}$ induced on $C^i_{\mathsf{bucket}} \subseteq A$ and $\bigcup_{\substack{x \in C^i_{\mathsf{bucket}} \\ j \in L}} N^{+j}(x) \subseteq B$. Our algorithm retains q_{th} variables in any non-empty C_{bucket}. Thus, $q_{\mathrm{th}} = |C^\ell_{\mathsf{bucket}}| < |C^i_{\mathsf{bucket}}|$. Due to our choice of q_{th}, Lemma 24 implies that a sunflower $\mathcal{P}$ of size q_{sf} exists in $G'_{\mathsf{bucket},L}$.

In the following two cases, we show the existence of a set $R^i \subseteq \mathcal{P} \subseteq C^{i+1}_{\mathsf{bucket}}$ of size k that satisfies the properties stated in the claim if the value of q_{sf} is sufficiently large. In particular, we show that

- **Case A.1:** If
$$q_{\mathrm{sf}} \geq (k + |\mathcal{V}^{\mathrm{neg}}|)r \cdot \frac{4kd}{\varepsilon^2} + k,$$
then we can find a set in $\mathcal{P}$ of size at least k that is compatible with S' with respect to colors in L.
- **Case A.2:** If
$$q_{\mathrm{sf}} \geq (k + |\mathcal{V}^{\mathrm{neg}}|)r(d-1)\left(\frac{2(k + |\mathcal{V}^{\mathrm{neg}}|)}{\varepsilon}\right)^d + k,$$
then again we can find a set in $\mathcal{P}$ of size at least k that is compatible with S' with respect to colors in H.

Since
$$q_{\mathrm{sf}} = (k + |\mathcal{V}^{\mathrm{neg}}|)r \cdot \frac{4kd}{\varepsilon^2} + (k + |\mathcal{V}^{\mathrm{neg}}|)r(d-1)\left(\frac{2(k + |\mathcal{V}^{\mathrm{neg}}|)}{\varepsilon}\right)^d + k + 1,$$
we first show the existence of a set $\mathcal{G}$ in $\mathcal{P}$ of size at least $(k + |\mathcal{V}^{\mathrm{neg}}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{\mathrm{neg}}|)}{\varepsilon}\right)^d + k + 1$ that contains variables that are compatible with S' with respect to colors in L. Then, we show that $\mathcal{G}$ contains a subset of size at least k that

is also compatible with S' with respect to colors in H, even after we delete y from $\mathcal{P}$. Thus, there will exist a set R^i in $\mathcal{P}$ of size at least k that is compatible with S' with respect to colors in both L and H. This will prove our claim for the case when variables in bucket appear in at least one clause of some color in L.

Case $A.1$**:** Consider the set of colors $j \in L$.

Recall that due to the application of Reduction Rule 11, $d^{+j}(z) \leq t_j$ for any $z \in C_{\mathsf{bucket}}$ and $j \in L$. As j is a color in L, the requirement for the color is $t_j < \frac{4kd}{\varepsilon^2}$. Hence, k variables in S' can be present in at most $k \sum_{j \in L} t_j \leq kr\frac{4kd}{\varepsilon^2}$ clauses in $\bigcup_{j \in L} \mathcal{C}_j$ as positive literals. Also, due to the application of Reduction Rule 11, $d^{-j}(z) \leq t_j$ for any $z \in \mathcal{V}^{neg}$. Thus, $|\mathcal{V}^{neg}|$ negative variables can be present in at most $|\mathcal{V}^{neg}| \sum_{j \in L} t_j \leq |\mathcal{V}^{neg}| r\frac{4kd}{\varepsilon^2}$ clauses in $\bigcup_{j \in L} \mathcal{C}_j$ as negative literals.

We know that the deleted variable y participates in $\mathcal{P}$. Note that $|\mathcal{P}| = q_{\text{sf}}$ and $\mathsf{petal}(u) \cap \mathsf{petal}(v) = \emptyset$ for distinct $u, v \in \mathcal{P}$. Since the number of clauses containing at least one negative literal in $\mathcal{V}^{neg}$ or variables from S' as a positive literal is bounded by $kr\frac{4kd}{\varepsilon^2} + |\mathcal{V}^{neg}| r\frac{4kd}{\varepsilon^2}$, it follows that there exists a set $\mathcal{G} \subseteq \mathcal{P}$ of size, say q_L, where

$$\begin{aligned} q_L &\geq q_{\text{sf}} - (k + |\mathcal{V}^{neg}|) r \frac{4kd}{\varepsilon^2} \\ &\geq (k + |\mathcal{V}^{\text{neg}}|) r(d-1) \left(\frac{2(k + |\mathcal{V}^{\text{neg}}|)}{\varepsilon} \right)^d + k + 1 \end{aligned}$$

such that none of the petals corresponding to variables in $\mathcal{G}$ contain any clause containing at least one negative literal or variables from S' as a positive literal. This implies for any $w \in \mathcal{G}$ and any $j \in L$,

$$\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w) \subseteq \mathsf{core}(\mathcal{P}) \tag{12.2}$$

This is because $N^{+j}(w) = \mathsf{petal}(w) \cup \mathsf{core}(\mathcal{P})$ and $\mathsf{sat}_{\Phi_j}(S') \cap \mathsf{petal}(w) = \emptyset$. Thus,

$$\begin{aligned}
|\mathsf{sat}_{\Phi_j}(S' \cup \{w\})| &= |\mathsf{sat}_{\Phi_j}(S')| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)| \\
&= |\mathsf{sat}_{\Phi_j}(S')| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap \mathsf{core}(\mathcal{P})| \\
&\qquad \text{(from Equation 12.2)} \\
&\geq |\mathsf{sat}_{\Phi_j}(S')| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(y)| \\
&\qquad \text{(since } \mathsf{core}(\mathcal{P}) \subseteq N^{+j}(y)\text{)} \\
&= |\mathsf{sat}_{\Phi_j}(S')| + |N^{+j}(y)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(y)| \\
&\qquad \text{(since } y, w \in C_{\mathsf{bucket}}\text{)} \\
&\geq |\mathsf{sat}_{\Phi_j}(S' \cup \{y\})| \\
&\geq t'_j \qquad \text{(since } y \text{ is compatible with respect to } j\text{)}
\end{aligned}$$

Thus, variables in $\mathcal{G}$ are potential candidates for our desired R^i as each of them is compatible with S' with respect to colors in L.

Case $A.2$**:** Now, we consider a color $j \in H$.

Recall from Definition 30 that a pair of variables u, v are called *heavy neighbors* of each other with respect to j if $|N^j(u) \cap N^j(v)| \geq \frac{\varepsilon t_j}{k+|\mathcal{V}^{neg}|}$. Thus, variables in S' can have at most $kr(d-1)\left(\frac{2(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^d$ heavy neighbors in $\mathcal{P}$ with respect to all $j \in [r]$, and variables in $\mathcal{V}^{neg}$ can have at most $|\mathcal{V}^{neg}|r(d-1)\left(\frac{2(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^d$ heavy neighbors in $\mathcal{P}$ with respect to all $j \in [r]$.

In the previous case (**Case** $A.1$), we showed the existence of a set $\mathcal{G}$ in $\mathcal{P}$, of size at least

$$(k + |\mathcal{V}^{\text{neg}}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{\text{neg}}|)}{\varepsilon}\right)^d + k + 1,$$

containing variables that are compatible with S' with respect to colors in L. Thus, there exists at least

$$|\mathcal{G}| - (k + |\mathcal{V}^{neg}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^d \geq k+1$$

variables in $\mathcal{P} \subseteq C^i_{\mathsf{bucket}}$, such that each variable w in the set is not a heavy neighbor of any variable in $\mathcal{V}^{neg} \cup S'$.

We now show that this set contains our desired R^i.

Case $A.2.1$ **:** $\mathsf{bucket}(j) = 0$**.** For any variable $u \in C_{\mathsf{bucket}}$, we know that $|N^{+j}(u)| \geq t_j$. Thus, replacing $x \in S \cap C_{\mathsf{bucket}}$ with any $w \in R^i$ yields $|N^{+j}(S - x + w)| \geq t_j$. Hence w is compatible with S' since $|\mathsf{sat}_{\Phi_j}(S' \cup \{w\})| \geq t_j \geq t'_j$.

Case $A.2.2$: $\mathsf{bucket}(j) \in [\lambda]$**.** For any pair of variables $u, v \in C_{\mathsf{bucket}}$, we have that $\frac{d^{+j}(u)}{(1+\varepsilon)} \leq d^{+j}(v) \leq (1+\varepsilon)d^{+j}(u)$. Thus, for any $w \in R^i$, we have

$$\begin{aligned}
|\mathsf{sat}_{\Phi_j}(S' \cup \{w\})| &= |\mathsf{sat}_{\Phi_j}(S')| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)| \\
&= |\mathsf{sat}_{\Phi_j}(S \setminus \{x\})| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)| \\
&\overset{(\spadesuit)}{\geq} |\mathsf{sat}_{\Phi_j}(S \setminus \{x\})| + |N^{+j}(w)| - (k - 1 + |\mathcal{V}^{neg}|) \cdot \frac{\varepsilon t_j}{k + |\mathcal{V}^{neg}|} \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - |N^{+j}(x)| + |N^{+j}(w)| - \varepsilon t_j \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - (1+\varepsilon)|N^{+j}(w)| + |N^{+j}(w)| - \varepsilon t_j \\
&\qquad\qquad \text{(since } x, w \in C_{\mathsf{bucket}} \text{ where } \mathsf{bucket}(j) \in [\lambda]) \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - \varepsilon|N^{+j}(w)| - \varepsilon t_j \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - 2\varepsilon t_j \qquad\qquad \text{(as } |N^{+j}(w)| \leq t_j)
\end{aligned}$$

($\spadesuit$) follows from the fact that w is not a neighbor of any vertex in $S' \cup \mathcal{V}^{neg}$.

Case $A.2.3$ **:** $\mathsf{bucket}(j) = \lambda + 1$**.** In this case, any variable $u \in C_{\mathsf{bucket}}$, we know that $|N^{+j}(u)| < \frac{\varepsilon t_j}{2k}$. Again, for any $w \in R^i$, we have

$$\begin{aligned}
|\mathsf{sat}_{\Phi_j}(S' \cup \{w\})| &= |\mathsf{sat}_{\Phi_j}(S')| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)| \\
&= |\mathsf{sat}_{\Phi_j}(S \setminus \{x\})| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)| \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - |N^{+j}(x)| + |N^{+j}(w)| - |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)| \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - |N^{+j}(x)| \;\; (\text{since } |N^{+j}(w)| \geq |\mathsf{sat}_{\Phi_j}(S') \cap N^{+j}(w)|) \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - \frac{\varepsilon t_j}{2k} \\
&\geq |\mathsf{sat}_{\Phi_j}(S)| - 2\varepsilon t_j
\end{aligned}$$

Case B: Now consider all buckets such that the variables in those bucket do not

appear as a positive literal in any clauses of color from L. In this case also, since

$$|C^{i+1}_{\mathsf{bucket}}| \geq q_{\text{th}} \geq (k+|\mathcal{V}^{neg}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^d + k + 1,$$

we have $R^i \subseteq C^{i+1}_{\mathsf{bucket}}$ such that each variable w in R^i satisfies the following: for all $v \in S' \cup \mathcal{V}^{neg}$ and for any $j \in [r]$, $|N^j(w) \cap N^j(v)| < \frac{\varepsilon t_j}{k+|\mathcal{V}^{neg}|}$. Thus, the analysis in **Cases** *A*.2.1**,** *A*.2.2 and *A*.2.3 hold here as well. □

Armed with Claim 9, we are ready to prove the statement via induction on the number of iterations. Let $z < \ell$ denote the least integer such that $x \notin C^z_{\mathsf{bucket}}$.

Base Case: This case corresponds to the z^{th} iteration. By the definition of z, it follows that $y \in C^{z-1}_{\mathsf{bucket}}$. Clearly, y is compatible with S'. By Claim 9, there exists a set $R^{z-1} \in C^{z-1}_{\mathsf{bucket}} \setminus \{y\}$ consisting of at least k variables such that each $w \in R^{z-1}$ is compatible with S'.

Inductive hypothesis. For some $z \leq i < \ell$, suppose that there exists a subset $R^i \subseteq C^i_{\mathsf{bucket}}$ of size at least k such that each variable in R^i is compatible with S'.

Inductive step. We show the existence of a set R^{i+1} of size at least k in $C^{i+1}_{\mathsf{bucket}}$. Note that the sets R^i and R^{i+1} may differ by at most one variable. First we consider the case that $R^i \subseteq C^{i+1}_{\mathsf{bucket}}$, i.e., the variable v that was deleted from C^i_{bucket} in order to obtain $C^{i+1}_{\mathsf{bucket}}$, does not belong to R^i. In this case, $R^{i+1} = R^i$ trivially satisfies the property via inductive hypothesis. Now, consider the case where the deleted variable v belongs to R^i. First, note that by the inductive hypothesis, v is compatible with S'. Therefore, we use Claim 9 with S, it follows that there exists a set $R' \subseteq \mathcal{P} \setminus \{v\} \subseteq C^i_{\mathsf{bucket}}$, such that $|R'| = k$, and each $w \in R'$ is compatible with S'. Since $|R^i| = k$ and $v \in R^i$, it follows that $R' \setminus (R^i \setminus \{v\}) \neq \emptyset$. Let $w' \in R' \setminus (R^i \setminus \{v\})$ be an arbitrary variable. It follows that $R^{i+1} = R^i - v + w'$ has size k, and each variable in R^{i+1} is compatible with S'. This finishes the inductive step. □

Lemma 27. *Suppose there exists some $S \subseteq \mathcal{V}$ of size k, such that for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(S)| \geq t_j$. Then, after exhaustive application of Reduction Rules 11 and 12, there exists some $S' \subseteq \mathcal{V}'$ of size k such that*

- *for each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S')| \geq \frac{t_j}{(1+\varepsilon)}$, and*
- *for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S')| \geq t_j$.*

Proof. Let $S = \{x_1, x_2, \ldots, x_k\}$. We will prove inductively that for each $0 \leq i \leq k$, there exists a set S_i satisfying the following properties.

1. $\{x_{i+1}, x_{i+2}, \ldots, x_k\} \subseteq S_i$, and $S_i \setminus \{x_{i+1}, x_{i+2}, \ldots, x_k\} \subseteq \mathcal{V}'$,
2. For each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S_i)| \geq t_j$, and
3. For each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S_i)| \geq t_j - 2i \cdot \varepsilon t_j$.

In the base case, let $S_0 \coloneqq S$ and note that S_0 trivially satisfies the required properties. Now suppose we have inductively shown the existence of a set S_{i-1} satisfying the aforementioned properties. We will show the existence of S_i. To this end, we consider cases based on whether $x_i \in \mathcal{V}'$.

Case 1: $x_i \in \mathcal{V}'$. In this case, let $S_i \coloneqq S_{i-1}$. Note that via inductive hypothesis and the fact that $x_i \in \mathcal{V}'$, the first property holds. Next, note that for each $j \in H$, by inductive hypothesis, $|\mathsf{sat}_{\Phi_j}(S_i)| = |\mathsf{sat}_{\Phi_j}(S_{i-1})| \geq t_j - 2(i-1)\varepsilon t_j \geq t_j - 2i\varepsilon t_j$. Finally, for each $j \in L$, via inductive hypothesis, $|\mathsf{sat}_{\Phi_j}(S_i)| = |\mathsf{sat}_{\Phi_j}(S_{i-1})| \geq t_j$.

Case 2: $x_i \notin \mathcal{V}'$. In this case, let $x_i \in C_{\mathsf{bucket}}$, and let $R_i \subseteq \widetilde{C}_{\mathsf{bucket}} \subseteq \mathcal{V}'^{pos}$ be a set of size k guaranteed by Lemma 26 satisfying the two properties mentioned in the statement thereof. Here, $\widetilde{C}_{\mathsf{bucket}} \subseteq C_{\mathsf{bucket}}$ is the *final subset* of the bucket that is retained after exhaustive application of the Reduction Rules 11 and 12. We first claim that $R_i \setminus S_{i-1} \neq \emptyset$. To this end, note that $|S_{i-1}| = |R_i| = k$, and $R_i \subseteq \widetilde{C}_{\mathsf{bucket}} \subseteq \mathcal{V}'^{pos}$. On the other hand, S_{i-1} contains at least one variable, namely x_i, such that $x_i \notin \mathcal{V}'^{pos}$. Therefore, $R_i \setminus S_{i-1} \neq \emptyset$. Let w be an arbitrary variable in $R_i \setminus S_{i-1}$. Let $S_i \coloneqq S_{i-1} - x_i + w$. Since $w \in \mathcal{V}'$, property (1) holds. Since w is compatible with $S_{i-1} \setminus \{x_i\}$, it follows that for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S_{i-1} \setminus \{x_i\} \cup \{w\})| \geq t_j$ due to inductive hypothesis. This shows property (2). We now show that S_i also satisfies property (3). To this end, consider any $j \in H$.

$$
\begin{aligned}
|\mathsf{sat}_{\Phi_j}(S_i)| &\geq \min\left\{t_j, |\mathsf{sat}_{\Phi_j}(S_{i-1})| - 2\varepsilon t_j\right\} && \text{(By property 1 of Lemma 26)} \\
&\geq |\mathsf{sat}_{\Phi_j}(S_{i-1})| - 2\varepsilon t_j \\
&\geq t_j - 2(i-1)\varepsilon t_j - 2\varepsilon t_j && \text{(By inductive hypothesis)} \\
&= t_j - 2i\varepsilon t_j.
\end{aligned}
$$

This completes the proof by induction.

Now, note that $S_k \subseteq \mathcal{V}'$ and we have

- for $j \in H$, $|\mathsf{sat}_{\Phi_j}(S_i)| = (1 - 2\varepsilon k)t_j \geq \frac{t_j}{(1+4\varepsilon k)}$, for $\varepsilon \in (0, \frac{1}{4k})$.
- for $j \in L$, $|\mathsf{sat}_{\Phi_j}(S_i)| \geq t_j$

Scaling ε to $\frac{\varepsilon}{4k}$, we get a set $S' = S_k$ satisfying the properties of Lemma 27.

Remark 8. *After scaling ε to $\frac{\varepsilon}{4k}$, the new bound on the set of positive variables is as follows:*

$$|\mathcal{V}^{'pos}| \leq \left(\frac{k\,(k + |\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}. \tag{12.3}$$

□

Lemma 28. *Let S'_{apx} denote any $(\beta_0, \beta_1, \ldots, \beta_r)$-approximate solution for the instance $\mathcal{I}'$. Then, S'_{apx} is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon))$-approximate solution for the instance $\mathcal{I}$.*

Proof. Let S_{opt} be an optimal solution to the instance $\mathcal{I}$. From Lemma 27, we know that there exists an $S' \subseteq \mathcal{V}'$ such that $|S'| = |S_{opt}|$ and

- for each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S')| \geq \frac{t_j}{(1+\varepsilon)}$, and
- for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S')| \geq t_j$.

Thus, S' is a feasible solution for $\mathcal{I}'$. Let S'_{opt} be an optimal solution for the reduced instance $\mathcal{I}'$. Then, we have,

$$|S'_{apx}| \leq \beta_0|S'_{opt}| \leq \beta_0|S'| \leq \beta_0|S_{opt}| \tag{12.4}$$

Also for each $j \in [r]$,

$$|\mathsf{sat}_{\Phi'_j}(S'_{apx})| \geq \frac{t_j}{\beta_j(1+\varepsilon)}$$

Since, $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{C}' \subseteq \mathcal{C}$, we have that for each $j \in [r]$,

$$|\mathsf{sat}_{\Phi_j}(S'_{apx})| \geq \frac{t_j}{\beta_j(1+\varepsilon)} \tag{12.5}$$

This concludes our proof. □

Proof of Lemma 23. Thus, the reduction algorithm $\mathcal{R}_{pos}$ along with the solution lifting algorithm Iden gives a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA. This, along with the size bound from Equation 12.3, proves Lemma 23. □

12.2.3 Bounding the number of clauses

The next step is to design a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA for F-Max k-weight SAT such that, given $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ as input, the reduction algorithm returns $\mathcal{I}' = ((\Phi' = (\mathcal{V}, \widetilde{\mathcal{C}}), r, f), k, t')$ where $|\widetilde{\mathcal{C}}|$ is bounded by a polynomial function of $|\mathcal{V}|$.

Lemma 29. *For any $\varepsilon \in (0, 1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA, say $(\mathcal{R}_C, \mathsf{Iden})$ for* F-Max k-weight SAT *when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free. Given an instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ of* F-Max k-weight SAT*, $size_{\mathcal{R}_C}(k) = |\mathcal{V}|^{\mathcal{O}(d)}$ and the formula returned by $\mathcal{R}_C$ is also $K_{d,d}$-free.*

Reduction algorithm. The reduction algorithm $\mathcal{R}_C$ takes an instance $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ of F-Max k-weight SAT as input and uses a process called *scaling and rounding of weights* to bound the set of clauses in $\mathcal{C}$ by a polynomial function of $|\mathcal{V}|$. It is described in Algorithm 13.

Algorithm 13 Procedure to bound the set of clauses $(\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t))$

Set $\widetilde{\mathcal{C}}$ be an empty multiset.
for each $j \in [r]$ **do**
  Let $s_j = \frac{\varepsilon t_j}{(1+\varepsilon)(d+1)|\mathcal{V}|^d}$.
  Let C_j^{set} denote the distinct clauses in $\mathcal{C}_j$.
  for each clause $c \in C_j^{\text{set}}$ **do**
    Let m_c^j denote the number of occurrences of c in C_j.
    Add $\lfloor m_c^j / s_j \rfloor$ copies of c to $\widetilde{C}$. We call this the normalization step.
  end for
end for
Let $\Phi' = (\mathcal{V}, \widetilde{C})$. Return the instance $\mathcal{I}' = ((\Phi', r, f), k, t')$, where $f' = f \restriction_{\widetilde{C}}$ and

$$t' = \left(\left\lceil \frac{t_1}{s_1(1+\varepsilon)} \right\rceil, \ldots, \left\lceil \frac{t_r}{s_r(1+\varepsilon)} \right\rceil \right).$$

We first prove that the reduction algorithm returns a reduced instance where the number of clauses is bounded by a polynomial function of the number of variables in the input formula.

Claim 10. *Let $\mathcal{I} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f), k, t)$ be an instance of* F-Max k-weight SAT*. Algorithm 13 bounds the number of clauses in the formula by $2(1 + \frac{1}{\varepsilon})(d+1)|\mathcal{V}|^{d+1}$.*

Proof. Let $\widetilde{C}_j$ denote the subset of $\widetilde{C}$ restricted to color class j. First, we show that, for every $j \in [r]$, $|\widetilde{C}^j|$ is bounded by some polynomial function of $|\mathcal{V}|$.

$$|\widetilde{C}^j| \leq \frac{|C_j|}{s_j} \leq \frac{2|\mathcal{V}| \cdot t_j}{s_j} \leq \frac{2|\mathcal{V}|(1+\varepsilon)(d+1)|\mathcal{V}|^d}{\varepsilon} = 2\left(1+\frac{1}{\varepsilon}\right)(d+1)|\mathcal{V}|^{d+1}$$

Here, $|C_j| \leq 2|\mathcal{V}|t_j$ due to Reduction Rule 11 and the fact that we can safely delete empty clauses. Thus, we have

$$|\widetilde{C}| \leq 2r\left(1+\frac{1}{\varepsilon}\right)(d+1)|\mathcal{V}|^{d+1}$$

□

Solution Lifting algorithm. The solution lifting algorithm lden takes as input the instance $\mathcal{I} = ((\Phi, r, f), k, t)$, the reduced instance $\mathcal{I}' = ((\Phi', r, f), k, t')$ which is the output of $\mathcal{R}_{\mathcal{C}}$ and a $(\beta_0, \beta_1, \ldots, \beta_r)$-approximate solution to the instance $\mathcal{I}'$, for some $(\beta_0, \beta_1, \ldots, \beta_r) \in \mathbb{N}^{r+1}$ and returns the approximate solution it receives as input.

Claim 11. *The number of distinct clauses in $\mathcal{C}$ is at most $(d+1)|\mathcal{V}|^d$.*

Proof. Let C^{set} denote the distinct clauses in $\mathcal{C}$. Since Φ is $K_{d,d}$-free, for every d-sized subset in $\mathcal{V}$ there can be at most $d-1$ common clauses in $\mathcal{C}$. Thus, the number of clauses of size at least d in $\mathcal{C}$ is bounded by $d \cdot \binom{|\mathcal{V}|}{d} \leq d|\mathcal{V}|^d$. Next, the number of clauses in $\mathcal{C}$ with distinct non-empty neighborhoods and size less than d is bounded by $\sum_{i=1}^{d} \binom{|\mathcal{V}|}{i} \leq |\mathcal{V}|^d$. Thus, we have $|C^{set}| \leq (d+1)|\mathcal{V}|^d$. □

Let S be an optimal solution for the original instance $\mathcal{I}$. We first claim that for every $j \in [r]$, we have $|\mathsf{sat}_{\Phi'_j}(S)| \geq \left\lceil \frac{t_j}{(1+\varepsilon)s_j} \right\rceil$, and hence S also satisfies all the constraint functions of the reduced instance.

Claim 12. *Let S be any set of k variables in $\mathcal{V}$, such that for any $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(S)| \geq t_j$. Then it holds that $|\mathsf{sat}_{\Phi'_j}(S)| \geq \left\lceil \frac{t_j}{s_j(1+\varepsilon)} \right\rceil$.*

Proof. Consider the variable set S satisfying the premise of the claim and fix an index $j \in [r]$ for the rest of the proof. Let $\mathsf{sat}_{\Phi_j}(S) = \mathcal{C}_1 \uplus \mathcal{C}_2 \uplus \cdots \uplus \mathcal{C}_\ell$ where for $i \in [\ell]$, $\mathcal{C}_i$ contains copies of same clause. We have $\ell \leq (d+1)|\mathcal{V}|^d$ due to Claim 11.

In $\mathsf{sat}_{\Phi'_j}(S)$, $\mathcal{C}_i$ now contributes $\left\lfloor \frac{|\mathcal{C}_i|}{s_j} \right\rfloor \geq \frac{|\mathcal{C}_i|}{s_j} - 1$. Therefore,

$$\begin{aligned}
|\mathsf{sat}_{\Phi'_j}(S)| \geq \sum_{i=1}^{\ell} \left\lfloor \frac{|\mathcal{C}_i|}{s_j} \right\rfloor &\geq \sum_{i=1}^{\ell} \left(\frac{|\mathcal{C}_i|}{s_j} - 1 \right) \\
&\geq \frac{t_j}{s_j} - \ell && \text{(Since } S \text{ is optimal for } I\text{)} \\
&\geq \frac{t_j}{s_j} - (d+1)|\mathcal{V}|^d \\
&= \frac{t_j}{\varepsilon t_j / ((1+\varepsilon)(d+1)|\mathcal{V}|^d)} - (d+1)|\mathcal{V}|^d \\
& && \text{(Using the definition of } s_j\text{)} \\
&= (d+1)|\mathcal{V}|^d \cdot \frac{1}{\varepsilon} \\
&= \frac{(d+1)|\mathcal{V}|^d}{\varepsilon} = \frac{t_j}{s_j(1+\varepsilon)},
\end{aligned}$$

Hence we have $|\mathsf{sat}_{\Phi'_j}(S)| \geq \frac{t_j}{s_j(1+\varepsilon)}$. However, since $|\mathsf{sat}_{\Phi'_j}(S)|$ is an integer, it must hold that $|\mathsf{sat}_{\Phi'_j}(S)| \geq \left\lceil \frac{t_j}{s_j(1+\varepsilon)} \right\rceil$. □

Note that Claim 12 implies that S is a feasible solution for the reduced instance of size k.

Lemma 30. *Let Y denote any $(\beta_0, \beta_1, \ldots, \beta_r)$-approximate solution for the instance $\mathcal{I}'$. Then, Y is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon))$-approximate solution for the instance $\mathcal{I}$.*

Proof. We have for every $j \in [r]$,

$$\begin{aligned}
|\mathsf{sat}_{\Phi_j}(Y)| &\geq s_j |\mathsf{sat}_{\Phi'_j}(Y)| \\
&\geq s_j \cdot \frac{1}{\beta_j} \left\lceil \frac{t_j}{s_j(1+\varepsilon)} \right\rceil \\
&\geq \frac{s_j}{\beta_j} \left(\frac{t_j}{s_j(1+\varepsilon)} \right) \\
&= \frac{t_j}{(1+\varepsilon)\beta_j}
\end{aligned}$$

Thus, Y satisfies all the constraint functions.

Let $Y^\star$ denote an optimal solution for the reduced instance. Since Y is a $(\beta_0, \beta_1, \ldots, \beta_r)$-

approximate solution for the reduced instance, we have

$$|Y| \leq \beta_0 |Y^\star| \leq \beta_0 |S|.$$

The last inequality follows from the fact S is a feasible solution of the reduced instance as shown in Claim 12. Hence Y is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon))$-approximate solution for the original instance $\mathcal{I}$. □

Hence, by Lemma 30, which proves that Iden is the desired solution lifting algorithm, and Claim 10, which proves that the number of clauses in the reduced instance is $2r(1+\frac{1}{\varepsilon})(d+1)n^{d+1}$ where n is the number of variables in the input formula, we obtain the desired $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA.

12.2.4 Putting together everything: an approximate kernel for F-Max k-weight SAT

We now give the prove of Theorem 19.

Proof. Given an instance $\mathcal{I} = ((\Phi, r, f), k, t)$, the reduction algorithm does the following.

- It first runs the reduction algorithm from $(1, 1+\frac{\varepsilon}{6}, \ldots, 1+\frac{\varepsilon}{6})$-APPA from Lemma 20.
- Then, it runs the reduction algorithm from $(1, 1+\frac{\varepsilon}{6}, \ldots, 1+\frac{\varepsilon}{6})$-APPA from Lemma 23.
- Finally it runs the the reduction algorithm from $(1, 1+\frac{\varepsilon}{6}, \ldots, 1+\frac{\varepsilon}{6})$-APPA from Lemma 29.

Thus, we get a $(1, (1+\frac{\varepsilon}{6})^3, \ldots, (1+\frac{\varepsilon}{6})^3)$-APPA, or equivalently a $(1, 1+\varepsilon, \ldots, 1+\varepsilon)$-APPA. This is because $\frac{1}{(1+\frac{\varepsilon}{6})^3} \geq \frac{1}{1+\varepsilon}$ for any $\varepsilon \in (0,1)$, and hence, for any $j \in [r]$, we have $\frac{t_j}{(1+\frac{\varepsilon}{6})^3} \geq \frac{t_j}{(1+\varepsilon)}$. Note that the first reduction algorithm only deletes clauses and hence if the input is a $K_{d,d}$-free formula, it returns a $K_{d,d}$-free formula as an output.

The first APPA returns a formula $\Phi' = (\mathcal{V}', \mathcal{C}')$ where the number of negative variables in $|\mathcal{V}'|$ is $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$. The second APPA bounds the number of positive variables

in $\mathcal{V}'$ by $\left(\frac{k(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2r)} = \left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^2r)}$. Finally, the third APPA bounds the number of clauses and the size of the reduced instance by $|\mathcal{V}|^{\mathcal{O}(d)} = \left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^3r)}$.

Thus the size of the reduced instance gets bounded by $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^3r)}$. This completes the proof of Theorem 19.

□

12.3 Lossy kernel for (M, F)-Max k-weight SAT

We give an EPSAKS for the (M, F)-Max k-weight SAT problem when parameterized by the solution size k. We formally define the problem below and then formulate it as a multi-criteria optimization problem.

(M, F)-Max k-weight SAT
Input: An instance $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$, where $\Phi = (\mathcal{V}, \mathcal{C})$ is a CNF-SAT formula, $f : \mathcal{C} \to [r]$ is a surjective function, and for each $j \in [r]$, we say that $\mathcal{C}_j := \{\mathfrak{s} \in \mathcal{C} : f(\mathfrak{s}) = j\}$ is the set of clauses of *color* j and, $t : [r] \to \mathbb{N}$ is a function such that $t_j := t(j) \geq 0$ is the *satisfiability requirement* for *color* j, for each $j \in [r]$. Moreover, we are given a matroid $\mathcal{M} = (\mathcal{V}, I)$ of rank k defined on the set of variables $\mathcal{V}$, where k is a non-negative integer.
Parameter: k
Question: Does there exist an assignment β of weight at most k to $\mathcal{V}$, such that $\beta^{-1}(1) \in I$, and for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(\beta)| \geq t_j$?
Here, $\mathsf{sat}_{\Phi_j}(\beta) \subseteq \mathcal{C}_j$ is the set of clauses of color $j \in [r]$ that are satisfied by β in the subformula $\Phi_j = (\mathcal{V}, \mathcal{C}_j)$.

Modeling as parameterized multi-criteria minimization problem: Given an instance $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$, where $\Phi = (\mathcal{V}, \mathcal{C})$, $\mathcal{M} = (\mathcal{V}, I)$ and $t = (t_1, \ldots, t_r)$, we model the above problem as a parameterized multi-criteria minimization problem as follows. For a given set of variables $S \subseteq \mathcal{V}$,

- the main optimization function w is defined as

$$w((\Phi, r, f, \mathcal{M}), k, S) = \min\{|S|, k+1\},$$

- the $r+1$ constraint functions are defined as:

  1. for $j \in [r]$,

$$\ell^j((\Phi, r, f, \mathcal{M}), k, S) = \begin{cases} -\infty & \text{if } |S| > k \\ |\mathsf{sat}_{\Phi_j}(S)| & \text{otherwise} \end{cases}$$

  2. for $j = r+1$,

$$\ell^j((\Phi, r, f, \mathcal{M}), k, S) = \begin{cases} 0 & \text{if } S \in I \\ -1 & \text{otherwise} \end{cases}$$

S is a solution to $((\Phi, r, f, \mathcal{M}), k, t)$ if, for all $j \in [r]$, $\ell^j((\Phi, r, f, \mathcal{M}), k, S) \geq t_j$, and $\ell^{r+1} \geq 0$.

Our goal is to design a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-approximate kernel, for every $\varepsilon > 0$, for (M, F)-Max k-weight SAT when the input formula Φ is $K_{d,d}$-free.

A brief overview of the approximate kernelization procedure: We design our approximate kernel in three main steps. The first step is to design an APPA that bounds the number of negative variables present in the formula by a polynomial function of k, ε. In the next step, we design an APPA that bounds the number of positive variables as a polynomial function of k, ε and preserves the set of negative variables. The third APPA bounds the number of clauses by a polynomial function of the number of variables in the formula. Combining the three APPAs gives our desired approximate kernel. We describe each step in the following three subsections and then prove the following theorem.

Theorem 20. *For any $\varepsilon \in (0,1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-approximate kernel for* (M, F)-Max k-weight SAT*, with $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(rd^2)}$ variables and $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(rd^3)}$ clauses, when the formula is $K_{d,d}$-free.*

12.3.1 Bounding the number of negative variables

We give a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA, for any $\varepsilon > 0$, for the (M, F)-Max k-weight SAT problem when the input CNF-SAT formula is $K_{d,d}$-free that bounds the number of negative variables.

Lemma 31. *For any $\varepsilon \in (0,1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA consisting of the pair $(\mathcal{R}_{neg}^{Mat}, \mathsf{Iden})$ for* (M, F)-Max k-weight SAT*, when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free. The reduction algorithm takes as input an instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M}), k, t)$ and outputs an instance $\mathcal{J}' = ((\Phi', r, f, \mathcal{M}), k, t')$ where $\Phi' = (\mathcal{V}', \mathcal{C}')$ is a $K_{d,d}$-free formula with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{C}' \subseteq \mathcal{C}$ and the number of negative variables in $\mathcal{V}'$ is at most $\mathcal{O}(\frac{rk^2}{\varepsilon})$.*

A brief outline of our APPA is as follows.

Reduction Algorithm: The reduction algorithm $\mathcal{R}_{\text{neg}}^{Mat}$ takes as input an instance $\mathcal{J}$ of (M, F)-Max k-weight SAT. It first applies Reduction Rule 13 exhaustively to remove any variable x such that $\{x\} \notin I$ as these variables can never participate in any solution. Then, it applies Reduction Rule 10 exhaustively to remove any clause with at least $k+1$ negative literals as such clauses will always be satisfied by any assignment of weight at most k. This is followed by the greedy procedure in Algorithm 12 that bounds the set of negative variables by a polynomial function of k and ε. The intuition behind the greedy procedure is that if the number of variables with high negative degree with respect to a color j is *large enough*, then any assignment of weight at most k satisfies at least t_j clauses of color class j, hence, we can forget clauses in the color class $\mathcal{C}_j$. Moreover, for any assignment of weight at most k to the variables, among the set of clauses containing any low negative degree, only a bounded number of clauses are unsatisfied. Here, by negative degree, we refer to the normalized negative degree, which is defined in Definition 28.

Solution Lifting Algorithm: The solution lifting algorithm Iden takes as input the instance $\mathcal{J}$, the output $\mathcal{J}'$ of $\mathcal{R}_{\text{neg}}^{Mat}$ which is called the reduced instance, and a $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution S' to $\mathcal{J}'$, for some $(\beta_0, \beta_1, \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$. It returns the approximate solution to the reduced instance that it receives as input.

Let $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$ be the input instance of (M, F)-Max k-weight SAT. We assume that the input matroid $\mathcal{M}$ has:

1. oracle access, and
2. rank k.

If $\mathsf{rank}(\mathcal{M}) > k$, then we implicitly work with the k-truncation $\mathcal{M}_k$; however for the sake of simplicity, we continue to use $\mathcal{M} = (\mathcal{V}, I)$ for the appropriately truncated

matroid.

Reduction Rule 13. *If there is a variable $x \in \mathcal{V}$ such that $\{x\} \notin I$, then do the following:*

- *Remove x from $\mathcal{V}$.*
- *For each $c \in \mathcal{C}$, if $x \in \mathsf{pos}(c)$, remove x from c.*
- *For each $c \in \mathcal{C}$, if $x \in \mathsf{neg}(c)$, remove c. If $f(c) = j$, then decrease t_j by 1.*

Return the reduced the instance $((\Phi' = (\mathcal{V}', \mathcal{C}'), r, f, \mathcal{M}), k, t')$.

Notice that Reduction Rule 13 is safe since such elements can never be part of any solution and can only be assigned 0.

Reduction Rule 10 is applied next which is also safe as it only removes clauses that are always satisfied by any assignment of weight at most k. After the application of this reduction rule, every clause in the reduced instance contains at most k negative literals. This is followed by applying the greedy procedure described in Algorithm 12 that bounds the number of negative variables $\mathcal{V}^{neg}$ in the formula. By Observation 10 and Claim 7, we know that, for every color $j \in [r]$, the algorithm

- either bounds the set of negative variables with high normalized negative degree with respect to color j by $\mathcal{O}(\frac{k^2}{\varepsilon})$, or
- shows that any assignment of weight at most k satisfies at least t_j clauses of color class j.

Note that the negative variables in the formula of the reduced instance are only the variables in Y_{high} since all the clauses with negative occurrences of variables in Y_{low} are deleted. Thus, the number of negative variables in the reduced instance is $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$ as shown in Observation 10.

Lemma 32. *Let S_{apx} be a $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution of the reduced instance $\mathcal{J}'$ for any $(\beta_0, \beta_1 \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$. Then, S_{apx} is a $(\beta_0, (1+\varepsilon)\beta_1, \ldots, (1+\varepsilon)\beta_r, \beta_{r+1})$-approximate solution of the input instance $\mathcal{J}$.*

Proof. Let S_{opt} be an optimal solution of the input instance $\mathcal{J}$. By Lemma 22, we know that $|S_{apx}| \leq \beta_0 |S_{opt}|$ and $\mathsf{sat}_{\Phi_j}(S_{apx}) \geq \frac{t_j}{\beta_j(1+\varepsilon)}$ for all $j \in [r]$.

Let S'_{opt} be an optimal solution of the reduced instance $\mathcal{J}'$. Then, we know that $S'_{opt} \in I$, and hence, $\ell^{r+1}(\mathcal{J}', k, S'_{opt}) = 0$. Thus, $\ell^{r+1}(\mathcal{J}', k, S_{apx}) \geq \frac{\ell^{r+1}(\mathcal{J}', k, S'_{opt})}{\beta_{r+1}} = 0$, implying that $S_{apx} \in I$. Since both the input and the reduced instances are defined over the same matroid $\mathcal{M}$[1], we have that $\ell^{r+1}(\mathcal{J}, k, S_{apx}) = 0 = \frac{\ell^{r+1}(\mathcal{J}, k, S_{opt})}{\beta_{r+1}}$. Thus, S_{apx} is a $(\beta_0, (1+\varepsilon)\beta_1, \ldots, (1+\varepsilon)\beta_r, \beta_{r+1})$-approximate solution of the input instance $\mathcal{J}$. □

Lemma 32 along with the fact that the number of negative variables in the reduced instance $\mathcal{J}'$ is $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$ proves Lemma 31.

12.3.2 Bounding the number of positive variables

In this subsection, we provide a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA for any $\varepsilon > 0$, for an instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M}), k, t)$ of (M, F)-Max k-weight SAT when the input formula Φ is $K_{d,d}$-free. This APPA bounds the size of $\mathcal{V}^{\text{pos}}$ by $f(k, \varepsilon, |\mathcal{V}^{neg}|)$ for some computable polynomial function f. Recall that for any $v \in \mathcal{V}^{pos}$, $d^-(v) = 0$. In particular, we prove the following lemma.

Lemma 33. *For any $\varepsilon \in (0, 1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA consisting of the pair $(\mathcal{R}^{Mat}_{pos}, \mathsf{Iden})$ for* (M, F)-Max k-weight SAT*, when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free. The reduction algorithm takes as input an instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M}), k, t)$ and outputs an instance $\mathcal{I}' = ((\Phi', r, f, \mathcal{M}), k, t')$ where $\Phi' = (\mathcal{V}', \mathcal{C}')$ is a $K_{d,d}$-free formula with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{C}' \subseteq \mathcal{C}$, the number of positive variables in $\mathcal{V}'$ is at most $\left(\frac{k(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}$, and $t'_j = \frac{t_j}{(1+\varepsilon)}$ for all $j \in [r]$. It preserves the set of negative variables in $\mathcal{V}$.*

A brief outline of our APPA is as follows.

Reduction Algorithm: The reduction algorithm $\mathcal{R}^{Mat}_{\text{pos}}$ takes as input an instance $\mathcal{J}$ of (M, F)-Max k-weight SAT. It does the following.

1. **Preliminary variable removal:** It applies a reduction rule that removes any variable x such that $\{x\} \notin I$ as these variables can never participate in any solution.

[1]Due to the hereditary property of the matroid, the subset of an independent set is also an independent set.

2. **Truncating Frequency:** We apply a reduction rule that bounds the frequency of every variable by at most $2t_j$.

3. **Bucketing Procedure:** We first partition the set of colors $[r]$ into two categories, based on their satisfiability requirements (high or low) and then implement the *bucketing procedure* to partition $\mathcal{V}^{pos}$ based on its *colored degree*. Each partition of $\mathcal{V}^{pos}$ under this procedure is called a *bucket*. The number of buckets created is a polynomial function of k and ε. Variables from $\mathcal{V}^{pos}$ in a bucket are present in *almost* the same number of clauses of any high requirement color j, and in exactly the same number of clauses of any low requirement color j.

4. **Filtering via Sunflower:** For each bucket created in the above step, we delete variables by finding a sufficiently large sunflower. Unlike the previous section, instead of arbitrarily deleting a variable participating in a sunflower, we cleverly use representative sets in order to carefully select a variable to delete. This bounds the size of each bucket by a polynomial function of k, ε and $|\mathcal{V}^{neg}|$.

Solution Lifting Algorithm: The solution lifting algorithm Iden takes as input the instance $\mathcal{J}$, the output $\mathcal{J}'$ of $\mathcal{R}_{\text{pos}}^{Mat}$ which is called the reduced instance, and a $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution S' to $\mathcal{J}'$, for some $(\beta_0, \beta_1, \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$. It returns the approximate solution to the reduced instance that it receives as input.

Let $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$ be the input instance of (M, F)-Max k-weight SAT. We assume that the input matroid $\mathcal{M}$ has:

1. oracle access, and

2. rank k. If $\mathsf{rank}(\mathcal{M}) > k$, then we implicitly work with the k-truncation of $\mathcal{M}$.

The Reduction Algorithm. We now describe the reduction algorithm $\mathcal{R}_{pos}$ in detail.

1. Preliminary variable removal: Reduction Rule 13 is applied to the instance. It is safe since any element x such that $\{x\} \notin I$ can never be part of any solution and can only be assigned 0.

2. Truncating Frequency: We implement Reduction Rule 11 to truncate the frequency of a variable in $\mathcal{V}$. After exhaustive application of this rule, the positive frequency as well as the negative frequency of every variable is at most t_j. This rule is safe as shown in Claim 8.

3. The Bucketing Procedure: We partition the set of colors $[r]$ into two categories, based on their satisfiability requirements.

- Set of colors with high satisfiability requirements, $H \coloneqq \left\{j \in [r] : t_j \geq \frac{4kd}{\varepsilon^2}\right\}$.

  We do *approximate* bucketing on $\mathcal{V}^{pos}$ with respect to the colors in H. Let $\lambda \coloneqq \lceil \log_{(1+\varepsilon)} \frac{2k}{\varepsilon} \rceil$. Fix a color $j \in H$. For each $\alpha \in [\lambda]$, we define

$$A(j, \alpha) \coloneqq \left\{ v \in \mathcal{V}^{\text{pos}} : \frac{t_j}{(1+\varepsilon)^{\alpha}} \leq d^{+j}(v) < \frac{t_j}{(1+\varepsilon)^{\alpha-1}} \right\}.$$

  We also define

$$A(j, 0) \coloneqq \left\{ v \in A : d^{+j}(v) \geq t_j \right\},$$

  and

$$A(j, \lambda+1) \coloneqq \left\{ v \in A : d^{+j}(v) < \frac{t_j}{(1+\varepsilon)^{\lambda}} \right\}.$$

- Set of colors with low satisfiability requirements, $L \coloneqq \left\{j \in [r] : t_j < \frac{4kd}{\varepsilon^2}\right\}$.

  We do *exact* bucketing on $\mathcal{V}^{pos}$ with respect to the colors in L. Fix a color $j \in L$. For each $\alpha \in \{0, 1, \ldots, t_j\}$, let $A(j, \alpha) \coloneqq \{v \in \mathcal{V}^{\text{pos}} : d^{+j}(v) = \alpha\}$.

Let $\mathcal{F}_{\text{b}}$ be the family of all possible functions $\mathsf{bucket} : [r] \to \mathbb{N}$ satisfying the following properties:

- for each $j \in H$, $\mathsf{bucket}(j) \in \{0, 1, 2, \ldots, \lambda+1\}$.
- for each $j \in L$, $\mathsf{bucket}(j) \in \{0, 1, 2, \ldots, t_j\}$.

Then, for each $\mathsf{bucket} \in \mathcal{F}_{\text{b}}$, we define a class $C_{\mathsf{bucket}} \coloneqq \bigcap_{j \in [r]} A(j, \mathsf{bucket}(j))$. Notice that $\mathcal{F}_{\text{b}}$ imposes a partitioning on $\mathcal{V}^{\text{pos}}$.

Observation 14. *For any pair of variables $u, v \in C_{\mathsf{bucket}}$, where C_{bucket} is a non-empty bucket of $\mathcal{V}^{pos}$ corresponding to some $\mathsf{bucket} \in \mathcal{F}_b$, we have the following:*

1. *For all $j \in H$ with $\mathsf{bucket}(j) \in [\lambda]$, $\frac{d^{+j}(v)}{(1+\varepsilon)} \leq d^{+j}(u) \leq (1+\varepsilon) d^{+j}(v)$.*

2. *For all* $j \in H$ *with* $\mathsf{bucket}(j) = 0$, *both* $d^{+j}(u)$ *and* $d^{+j}(v)$ *are at least* t_j.

3. *For all* $j \in H$ *with* $\mathsf{bucket}(j) = \lambda + 1$, *both* $d^{+j}(u)$ *and* $d^{+j}(v)$ *are at most* $\frac{\varepsilon t_j}{2k}$.

4. *For all* $j \in L$, $d^{+j}(u) = d^{+j}(v)$.

The total number of buckets returned by the bucketing procedure is $\mathcal{O}\left(\left(\frac{kd}{\varepsilon^2}\right)^r\right)$.

4. Filtering via Sunflower: For each non-empty C_{bucket} of $\mathcal{V}^{\text{pos}}$, if the number of variables in it is more than $q_{\text{th}}^{Mat} := d((q_{\text{sf}}^{Mat} - 1)r\frac{4kd}{\varepsilon^2})^d$, where

$$\begin{aligned} q_{\text{sf}}^{Mat} &:= k \cdot q_{\text{sf}} + 1 \\ &= k\left((k + |\mathcal{V}^{neg}|)r\frac{4kd}{\varepsilon^2} + (k + |\mathcal{V}^{neg}|)r(d-1)\left(\frac{2(k + |\mathcal{V}^{neg}|)}{\varepsilon}\right)^d + k + 1\right) + 1, \end{aligned}$$

we apply Reduction Rule 14 described below.

For a given formula $\Phi = (\mathcal{V}, \mathcal{C})$, let $G_{\mathsf{bucket},L} = (A, B)$ be the incidence bipartite graph constructed as follows.

- For every $v \in C_{\mathsf{bucket}}$, add a vertex a_v to A.
- For every clause $c \in \mathcal{C}$ such that $f(c) \in L$, add a vertex b_c to B.
- Add an edge between a_v and b_c iff $c \in N(v)$.
- Delete all isolated vertices.

Since the formula Φ is $K_{d,d}$-free, $G_{\mathsf{bucket},L}$ is also $K_{d,d}$-free as it is the incidence graph of a subformula. According to the construction, for a sunflower $\mathcal{P} \subseteq A$, the sets correspond to neighborhoods of vertices from A participating in $\mathcal{P}$, i.e., the family $\{N(a_v) \mid a_v \in \mathcal{P}\}$. The *core* Y of $\mathcal{P}$ represents the set of vertices in B that are present in the neighborhoods of each vertex a_v that participates in $\mathcal{P}$, i.e., all the vertices in $\bigcap_{a_v \in \mathcal{P}} N(a_v)$. Similarly, for any vertex $a_v \in \mathcal{P}$, the *petal* corresponding to a_v, denoted by $\mathsf{petal}(a_v)$, is $N(a_v) \setminus Y$.

In the language of CNF-SAT formulae, the sets in the sunflower $\mathcal{P} \subseteq A$ correspond to the family of clauses $\{\mathcal{C}[v] \mid a_v \in \mathcal{P}\}$, where $\mathcal{C}[v] = N^+(v) \bigcap (\cup_{j \in L} C_j)$, i.e. $\mathcal{C}[v]$ is the set of clauses from color classes in L that contain v. Notice that as the *bucketing procedure* is implemented only on $\mathcal{V}^{pos}$, $N(v) = N^+(v)$ for any $v \in \mathcal{V}^{pos}$. The *core* Y

of $\mathcal{P}$ represents the set of clauses that contain each variable v corresponding to the vertex $a_v \in A$ that participates in $\mathcal{P}$, i.e., all the clauses in $\bigcap_{a_v \in \mathcal{P}} \mathcal{C}[v]$. Similarly, for any variable $v \in C_{\mathsf{bucket}}$, if $a_v \in \mathcal{P}$, then the *petal* denoted by $\mathsf{petal}(a_v)$ is $\mathcal{C}[v] \setminus Y$. Thus, the sunflower $\mathcal{P}$ collects all variables in C_{bucket} such that for any pair $u, v \in C_{\mathsf{bucket}}$, if $a_u, a_v \in \mathcal{P}$, then $\mathcal{C}[u] \cap \mathcal{C}[v]$ is exactly the same.

Due to exhaustive application of Reduction Rule 11, the positive frequency of every variable is at most t_j when restricted to clauses with color j. Thus, the degree of every vertex in A in the incidence graph is bounded by $\sum_{j \in L} t_j$. Since $j \in L$ we have $t_j \leq \frac{4kd}{\varepsilon^2}$ which implies $\sum_{j \in L} t_j \leq r\frac{4kd}{\varepsilon^2}$. If $|C_{\mathsf{bucket}}| > q_{\mathrm{th}}^{Mat}$, then using Lemma 24 with $a = b = d$, $\ell = r\frac{4kd}{\varepsilon^2}$ and $w = q_{\mathrm{sf}}^{Mat}$, a sunflower $\mathcal{P}$ in $G_{\mathsf{bucket},L}$ of size q_{sf}^{Mat} exists.

Reduction Rule 14 (Matroidal Sunflower rule)**.** *Given an instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M}), k, t)$, for each $\mathsf{bucket} \in \mathcal{F}_b$, if $|C_{\mathsf{bucket}}| > q_{th}^{Mat}$, then we apply the reduction rule based on two cases:*

1. *if for $v \in C_{\mathsf{bucket}}$, there exists $j \in L$ such that $d^{+j}(v) > 0$, then we compute a sunflower $\mathcal{P} \subseteq C_{\mathsf{bucket}}$ of size q_{sf}^{Mat}. Let $\mathcal{P}_0 := \mathcal{P}$. We perform the greedy procedure listed below.*
2. *if for $v \in C_{\mathsf{bucket}}$, and for all $j \in L$, $d^{+j}(v) = 0$, then let $\mathcal{P}_0 := C_{\mathsf{bucket}}$. We perform the greedy procedure listed below.*

Greedy procedure:

1. *For each $1 \leq \iota \leq q_{sf}$, we compute $Q_\iota \subseteq_{rep}^{k-1} \mathcal{P}_{\iota-1}$, and let $\mathcal{P}_\iota := \mathcal{P}_{\iota-1} \setminus Q_\iota$.*
2. *Let v be a variable in $\mathcal{P}_{q_{sf}}$. We update C_{bucket} by deleting v.*

After exhaustively applying the above procedure, return the instance $\mathcal{J}' = ((\Phi' = (\mathcal{V}', \mathcal{C}'), r, f, \mathcal{M}), k, t')$ where $\mathcal{V}'$ is the set of variables left, $\mathcal{C}' \subseteq \mathcal{C}$ where $\mathcal{C}'$ is the set of clauses restricted to $\mathcal{V}'$, and set $t'_j = \frac{t_j}{1+\varepsilon}$ for all $j \in [r]$.

Observation 15. *Due to the exhaustive application of Reduction Rule 13, each $u \in \mathcal{V}$ forms a singleton independent set in I, which implies that $|Q_\iota| \geq 1$ for each $\iota \in [q_{sf}]$. Further, due to Lemma 1, Q_ι, for any $\iota \in [q_{sf}]$, can be computed in polynomial time using polynomially many calls to the independence oracle for $\mathcal{M}$.*

Remark 9. *Note that while we do not actively delete clauses, a clause $c \in \mathcal{C}$ may inadvertently get deleted if all the variables appearing in it get deleted during the application of Reduction Rules 13, 11 or 14.*

Observation 16. *Due to the construction of C_{bucket}, only variables from $\mathcal{V}^{pos}$ are being deleted. Thus, the set of negative variables $\mathcal{V}^{neg}$ is preserved by the algorithm.*

The reduction algorithm $\mathcal{R}^{Mat}_{pos}$ partitions $\mathcal{V}^{\text{pos}}$ into $\mathcal{O}\left(\left(\frac{kd}{\varepsilon^2}\right)^r\right)$ buckets and each bucket contains no more than q^{Mat}_{th} variables. Hence, the size of $\mathcal{V}^{\text{pos}}$ in the reduced instance is bounded. We have

$$|\mathcal{V}'^{pos}| = \mathcal{O}\left(\left(\frac{kd}{\varepsilon^2}\right)^r\right) \cdot q^{Mat}_{\text{th}} \leq \left(\frac{k + |\mathcal{V}^{neg}|}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}. \tag{12.6}$$

Solution lifting algorithm We describe the solution lifting algorithm Iden as follows. It takes as input the instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M}), k, t)$, the reduced instance $\mathcal{J}' = ((\Phi' = (\mathcal{V}', \mathcal{C}'), r, f, \mathcal{M}), k, t')$ which is the output of the reduction algorithm $\mathcal{R}^{Mat}_{pos}$, and S' which is a $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution to $\mathcal{J}'$, for any $(\beta_0, \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$, where $\beta_i > 1$ for all $i \in [0, r+1]$. It simply returns the solution S' it receives as input. In the following discussion, we show that S' is a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon), \beta_{r+1})$-approximate solution to $\mathcal{J}$, thus establishing $(\mathcal{R}^{Mat}_{pos}, \mathsf{Iden})$ as a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA.

Lemma 34. *Let $\mathcal{M} = (\mathcal{V}, I)$ be the given input matroid. Suppose there exists some $S \subseteq \mathcal{V}$ of size k, such that $S \in I$ and for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(S)| \geq t'_j$, where $0 \leq t'_j \leq t_j$. Consider a C_{bucket} for some $\mathsf{bucket} \in \mathcal{F}_b$ such that $S \cap C_{\mathsf{bucket}} \neq \emptyset$. Let $x \in (S \cap C_{\mathsf{bucket}})$ and $S' = S \setminus \{x\}$. Let $C'_{\mathsf{bucket}} \subseteq C_{\mathsf{bucket}}$ be the* final subset *of the bucket that is retained after repeated application of the matroidal sunflower rule. Then, at least one of the following holds:*

- *$x \in C'_{\mathsf{bucket}}$, or*
- *there exists a set $R \subseteq C'_{\mathsf{bucket}}$ of size k such that for each $w \in R$ the following properties hold.*
  1. *For each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S - x + w)| \geq \min\left\{t'_j, |\mathsf{sat}_{\Phi_j}(S)| - 2\varepsilon t_j\right\}$,*
  2. *For each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S - x + w)| \geq t'_j$, and*
  3. *$S - x + w \in I$.*

Proof. At a high level, the proof strategy is analogous to that of Lemma 26, although here we also have to additionally maintain compatibility w.r.t. matroid independence.

We refer to each application of the procedure in the sunflower rule (Reduction Rule 14) on C_{bucket} as an *iteration*. Let $C^i_{\mathsf{bucket}} \subseteq C_{\mathsf{bucket}}$ be the subset of C_{bucket} after i iterations, where the total number of iterations is $\ell \geq 0$. Note that if $x \in C^\ell_{\mathsf{bucket}} = \widetilde{C}_{\mathsf{bucket}}$, then the statement of the lemma is obviously true. In the rest of the proof, we consider the case when $x \notin C^\ell_{\mathsf{bucket}}$. Let $S' = S \setminus \{x\}$. We will say that a variable $y \in C_{\mathsf{bucket}}$ is *compatible with* S' if it satisfies the following two properties (analogous to properties (1), (2) and (3) in the statement of the lemma):

(a) for each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S' \cup \{y\})| \geq \min\left\{t'_j, |\mathsf{sat}_{\Phi_j}(S)| - 2\varepsilon t_j\right\}$

(b) for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S' \cup \{y\})| \geq t'_j$.

(c) $S' \cup \{y\} \in I$.

Now, we prove the following analogue of Claim 9, which is the only crucial ingredient to complete the proof by induction.

Claim 13. *Let $y \in C^i_{\mathsf{bucket}}$ be a vertex that is compatible with S', and suppose y is deleted from C^i_{bucket} by an application of Reduction Rule 14 (the matroidal sunflower rule) on sunflower $\mathcal{P}$, in order to obtain $C^{i+1}_{\mathsf{bucket}} = C^i_{\mathsf{bucket}} \setminus \{y\}$. Then there exists a subset $R^i \subseteq \mathcal{P} \setminus \{y\}$ of size k, such that each $w \in R^i$ is compatible with S'.*

Proof. Since y is deleted by an application of Reduction Rule 14, we have that $y \in \mathcal{P}_{q_{\mathrm{sf}}}$. This implies that $y \notin Q_1 \cup Q_2 \cup \ldots \cup Q_{q_{\mathrm{sf}}}$, which also implies that $y \in \mathcal{P}_{q_{\mathrm{sf}}} \subset \mathcal{P}_{q_{\mathrm{sf}}-1} \subset \ldots \subset \mathcal{P}_1 \subset \mathcal{P}_0$. The strict subset relation among the $\mathcal{P}_j$s, for $j \in [0, q_{\mathrm{sf}}]$, is due to the fact that each $|Q_\iota| \geq 1$, for $\iota \in [q_{\mathrm{sf}}]$, as noted in Observation 15. We know that y is compatible with S'. Thus, it follows that, for each $1 \leq \iota \leq q_{\mathrm{sf}}$, $\mathcal{P}_{\iota-1}$ contains a variable, specifically y, such that $S' \cup \{y\} \in I$. Therefore, for each $1 \leq \iota \leq q_{\mathrm{sf}}$, Q_ι must contain a variable z_ι such that $S' \cup \{z_\iota\} \in I$ because $Q_\iota \subseteq^{k-1}_{rep} \mathcal{P}_{\iota-1}$. Notice that that size of each Q_ι, for $\iota \in [q_{\mathrm{sf}}]$, is between 1 and k since it is a $(k-1)$-representative set of $\mathcal{P}_\iota$. Moreover, $Q_a \cap Q_b = \emptyset$ for any $a \neq b \in [q_{\mathrm{sf}}]$.

Case A: Consider a bucket for which there exists at least one color $j \in L$ such that the variables in that bucket appear as positive literals in some clause of color j.

By following arguments for **Case** A in Claim 9, as

$$q_{\mathrm{sf}} \geq (k + |\mathcal{V}^{\mathrm{neg}}|)r \cdot \frac{4kd}{\varepsilon^2} + (k + |\mathcal{V}^{\mathrm{neg}}|)r(d-1)\left(\frac{2(k + |\mathcal{V}^{\mathrm{neg}}|)}{\varepsilon}\right)^d + k,$$

it follows that, there exist at least k distinct indices $1 \leq \iota_1 < \iota_2 < \ldots < \iota_k \leq q_{\text{sf}}$, such that for each $1 \leq \jmath \leq k$, and for all $z \in Q_{\iota_\jmath}$, $S' \cup \{z\}$ satisfies properties (a) and (b).

Case B: Now consider all buckets such that the variables in those bucket do not appear as a positive literal in any clauses of color from L. Similarly, by following arguments for **Case** B in Claim 9, as

$$q_{\text{sf}} \geq (k + |\mathcal{V}^{\text{neg}}|)r(d-1)\left(\frac{2(k+|\mathcal{V}^{\text{neg}}|)}{\varepsilon}\right)^d + k,$$

there exist at least k distinct indices $1 \leq \iota_1 < \iota_2 < \ldots < \iota_k \leq q_{\text{sf}}$, such that for each $1 \leq \jmath \leq k$, and for all $z \in Q_{\iota_\jmath}$, $S' \cup \{z\}$ satisfies properties (a) and (b).

Finally, in both the cases, note that each $Q_{\iota_\jmath}$ contains at least one vertex z' such that $S' \cup \{z'\} \in I$. This is due to the fact that $S' \cup \{y\} \in I$, $y \in \mathcal{P}_{\iota_\jmath - 1}$ and $Q_{\iota_\jmath} \subseteq^{k-1}_{rep} \mathcal{P}_{\iota_\jmath - 1}$. Therefore, for each $1 \leq \jmath \leq k$, there exists a vertex $z' \in Q_{\iota_\jmath}$ such that z' is compatible with S'. □

We now prove the statement via induction on the number of iterations. Let $z < \ell$ denote the least integer such that $x \notin C^z_{\mathsf{bucket}}$.

Base Case: This case corresponds to the z^{th} iteration. By the definition of z, it follows that $y \in C^{z-1}_{\mathsf{bucket}}$. Clearly, y is compatible with S'. By Claim 13, there exists a set $R^{z-1} \in C^{z-1}_{\mathsf{bucket}} \setminus \{y\}$ consisting of at least k variables such that each $w \in R^{z-1}$ is compatible with S'.

Inductive hypothesis. For some $z \leq i < \ell$, suppose that there exists a subset $R^i \subseteq C^i_{\mathsf{bucket}}$ of size at least k such that each variable in R^i is compatible with S'.

Inductive step. We show the existence of a set R^{i+1} of size at least k in $C^{i+1}_{\mathsf{bucket}}$. Note that the sets R^i and R^{i+1} may differ by at most one variable. First we consider the case that $R^i \subseteq C^{i+1}_{\mathsf{bucket}}$, i.e., the variable v that was deleted from C^i_{bucket} in order to obtain $C^{i+1}_{\mathsf{bucket}}$, does not belong to R^i. In this case, $R^{i+1} = R^i$ trivially satisfies the property via inductive hypothesis. Now, consider the case where the deleted variable v belongs to R^i. First, note that by the inductive hypothesis, v is compatible with S'. Therefore, we use Claim 13 with S, it follows that there exists a set $R' \subseteq \mathcal{P} \setminus \{v\} \subseteq C^i_{\mathsf{bucket}}$, such that $|R'| = k$, and each $w \in R'$ is compatible with S'. Since $|R^i| = k$ and $v \in R^i$, it follows that $R' \setminus (R^i \setminus \{v\}) \neq \emptyset$. Let $w' \in R' \setminus (R^i \setminus \{v\})$ be an arbitrary variable. It follows that $R^{i+1} = R^i - v + w'$ has size k, and each

variable in R^{i+1} is compatible with S'. This finishes the inductive step.

□

Lemma 35. *Let $\mathcal{M} = (\mathcal{V}, I)$ be the given input matroid of rank k. Suppose there exists some $S \subseteq \mathcal{V}$ of size k, such that $S \in I$, and for each $j \in [r]$, $|\mathsf{sat}_{\Phi_j}(S)| \geq t_j$. Then, there exists some $S' \subseteq \mathcal{V}'$ such that,*

- *$S' \in I$,*
- *for each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S')| \geq \frac{t_j}{(1+\varepsilon)}$, and*
- *for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S')| \geq t_j$.*

Proof. Let $S = \{x_1, x_2, \ldots, x_k\}$. We will prove inductively that for each $0 \leq i \leq k$, there exists a set S_i of size k satisfying the following properties.

1. $\{x_{i+1}, x_{i+2}, \ldots, x_k\} \subseteq S_i$, and $S_i \setminus \{x_{i+1}, x_{i+2}, \ldots, x_k\} \subseteq \mathcal{V}'$,
2. For each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S_i)| \geq t_j$,
3. For each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S_i)| \geq t_j - 2i \cdot \varepsilon t_j$, and
4. $S_i \in I$.

In the base case, let $S_0 \coloneqq S$ and note that S_0 trivially satisfies the required properties. Now suppose we have inductively shown the existence of a set S_{i-1} satisfying the aforementioned properties. We will show the existence of S_i. To this end, we consider cases based on whether $x_i \in \mathcal{V}'$.

Case 1: $x_i \in \mathcal{V}'$. In this case, let $S_i \coloneqq S_{i-1}$. Note that via inductive hypothesis and the fact that $x_i \in \mathcal{V}'$, the first property holds. Next, note that for each $j \in H$, by inductive hypothesis, $|\mathsf{sat}_{\Phi_j}(S_i)| = |\mathsf{sat}_{\Phi_j}(S_{i-1})| \geq t_j - 2(i-1)\varepsilon t_j \geq t_j - 2i\varepsilon t_j$. Finally, for each $j \in L$, via inductive hypothesis, $|\mathsf{sat}_{\Phi_j}(S_i)| = |\mathsf{sat}_{\Phi_j}(S_{i-1})| \geq t_j$. Moreover, we have $S_i \in I$ as $S_i = S_{i-1}$ and by induction hypothesis, $S_{i-1} \in I$.

Case 2: $x_i \notin \mathcal{V}'$. In this case, let $x_i \in C_{\mathsf{bucket}}$, and let $R_i \subseteq \widetilde{C}_{\mathsf{bucket}} \subseteq \mathcal{V}'^{pos}$ be a set of size k obtained by Lemma 34 satisfying the three properties mentioned in the statement thereof. Here, $\widetilde{C}_{\mathsf{bucket}} \subseteq C_{\mathsf{bucket}}$ is the *final subset* of the bucket that is retained after repeated application of the matroidal sunflower rule. We first claim that $R_i \setminus S_{i-1} \neq \emptyset$. To this end, note that $|S_{i-1}| = |R_i| = k$, and $R_i \subseteq \widetilde{C}_{\mathsf{bucket}} \subseteq \mathcal{V}'^{pos}$. On the other hand, S_{i-1} contains at least one variable, namely x_i, such that

$x_i \notin \mathcal{V}'^{pos}$. Therefore, $R_i \setminus S_{i-1} \neq \emptyset$. Let w be an arbitrary variable in $R_i \setminus S_{i-1}$. Let $S_i \coloneqq S_{i-1} - x_i + w$. Since $w \in \mathcal{V}'$, property (1) holds. Since w is compatible with $S_{i-1} \setminus \{x_i\}$, it follows that for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S_{i-1} \setminus \{x_i\} \cup \{w\})| \geq t_j$ due to inductive hypothesis. This shows property (2). We now show that S_i also satisfies property (3). To this end, consider any $j \in H$.

$$\begin{aligned}
|\mathsf{sat}_{\Phi_j}(S_i)| &\geq \min\left\{t_j, |\mathsf{sat}_{\Phi_j}(S_{i-1})| - 2\varepsilon t_j\right\} && \text{(By property 1 of Lemma 34)} \\
&\geq |\mathsf{sat}_{\Phi_j}(S_{i-1})| - 2\varepsilon t_j \\
&\geq t_j - 2(i-1)\varepsilon t_j - 2\varepsilon t_j && \text{(By inductive hypothesis)} \\
&= t_j - 2i\varepsilon t_j.
\end{aligned}$$

For each $y \in R_i$, we have $S_{i-1} - x_i + y \in I$. It follows that S_i satisfies $S_i \subseteq \mathcal{V}'$, and $S_i \in I$ as $w \in R_i$.

Now, note that $S_k \subseteq \mathcal{V}'$ and we have

- for $j \in H$, $|\mathsf{sat}_{\Phi_j}(S_k)| = (1 - 2\varepsilon k)t_j \geq \frac{t_j}{(1+4\varepsilon k)}$, for $\varepsilon \in (0, \frac{1}{4k})$,
- for $j \in L$, $|\mathsf{sat}_{\Phi_j}(S_k)| \geq t_j$, and
- $S_k \in I$.

Scaling ε to $\frac{\varepsilon}{4k}$, we get a set $S' = S_k$ satisfying the properties of Lemma 35.

Remark 10. *After scaling ε to $\frac{\varepsilon}{4k}$, the new bound on the set of positive variables is as follows:*

$$|\mathcal{V}'^{pos}| \leq \left(\frac{k\,(k + |\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2+r)}. \tag{12.7}$$

□

Lemma 36. *For any $(\beta_0, \beta_1, \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$, let S'_{apx} denote any $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution for the instance $\mathcal{J}'$. Then, S'_{apx} is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon), \beta_{r+1})$-approximate solution for the instance $\mathcal{J}$.*

Proof. Let S_{opt} be an optimal solution to the instance $\mathcal{J}$. From Lemma 35, we know that there exists an $S' \subseteq \mathcal{V}'$ such that $|S'| = |S_{opt}|$ and

- for each $j \in H$, $|\mathsf{sat}_{\Phi_j}(S')| \geq \frac{t_j}{(1+\varepsilon)}$,
- for each $j \in L$, $|\mathsf{sat}_{\Phi_j}(S')| \geq t_j$, and

- $S' \in I$.

Thus, S' is a feasible solution for $\mathcal{J}'$. Let S'_{opt} be an optimal solution for the reduced instance $\mathcal{J}'$. Then, we have,

$$|S'_{apx}| \leq \beta_0 |S'_{opt}| \leq \beta_0 |S'| \leq \beta_0 |S_{opt}| \tag{12.8}$$

Also, for each $j \in [r]$,

$$|\mathsf{sat}_{\Phi'_j}(S'_{apx})| \geq \frac{t_j}{\beta_j(1+\varepsilon)}$$

Since, $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{C}' \subseteq \mathcal{C}$, we have that for each $j \in [r]$,

$$|\mathsf{sat}_{\Phi_j}(S'_{apx})| \geq \frac{t_j}{\beta_j(1+\varepsilon)} \tag{12.9}$$

Since $S'_{opt} \in I$, we have, $\ell^{r+1}(\mathcal{J}', k, S'_{opt}) = 0$. Thus, $\ell^{r+1}(\mathcal{J}', k, S'_{apx}) \geq \frac{\ell^{r+1}(\mathcal{J}', k, S'_{opt})}{\beta_{r+1}} = 0$, implying that $S'_{apx} \in I$. Since both the input and the reduced instances are defined over the same matroid $\mathcal{M}$[2], we have that $\ell^{r+1}(\mathcal{J}, k, S'_{apx}) = 0 = \frac{\ell^{r+1}(\mathcal{J}, k, S_{opt})}{\beta_{r+1}}$. By Equation 12.9, we know that S'_{apx} satisfies all the other constraint functions for $j \in [r]$.

Hence S'_{apx} is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon), \beta_{r+1})$-approximate solution for the original instance $\mathcal{J}$.

□

Proof of Lemma 33. Thus, the reduction algorithm $\mathcal{R}^{Mat}_{pos}$ along with the solution lifting algorithm $\mathsf{Ide}n$ gives a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA. This, along with the size bound from Equation 10, proves Lemma 33. □

12.3.3 Bounding the number of clauses

We design a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA for (M, F)-Max k-weight SAT such that, given $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M} = (\mathcal{V}, I)), k, t)$ as input, the reduction algorithm returns $\mathcal{J}' = ((\Phi' = (\mathcal{V}, \widetilde{\mathcal{C}}), r, f, \mathcal{M}), k, t')$ where $\widetilde{\mathcal{C}} \subseteq \mathcal{C}$ and $|\widetilde{\mathcal{C}}|$ is bounded by a polynomial function of $|\mathcal{V}|$.

[2]Due to the hereditary property of the matroid, the subset of an independent set is also an independent set.

Lemma 37. *For any $\varepsilon \in (0,1)$, there exists a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA, say $(\mathcal{R}_C^{Mat}, \mathsf{Iden})$ for* (M, F)-Max k-weight SAT *when the input formula $\Phi = (\mathcal{V}, \mathcal{C})$ is $K_{d,d}$-free. Given an input instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M}), k, t)$, $size_{\mathcal{R}_C^{Mat}}(k) = |\mathcal{V}|^{\mathcal{O}(d)}$ and the formula returned by $\mathcal{R}_C^{Mat}$ is also $K_{d,d}$-free.*

Let $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$ be the input instance of (M, F)-Max k-weight SAT. We assume that the input matroid $\mathcal{M}$ has:

1. oracle access, and
2. rank k. If $\mathsf{rank}(\mathcal{M}) > k$, then we implicitly work with the k-truncation of $\mathcal{M}$.

This APPA does not modify the set of variables. Hence, the matroid constraint remains unaffected.

Reduction algorithm. The reduction algorithm $\mathcal{R}_C^{Mat}$ runs Algorithm 13 on an instance $\mathcal{J} = ((\Phi = (\mathcal{V}, \mathcal{C}), r, f, \mathcal{M} = (\mathcal{V}, I)), k, t)$ of (M, F)-Max k-weight SAT. It uses a process called *scaling and rounding of weights* to bound the set of clauses in $\mathcal{C}$ by a polynomial function of $|\mathcal{V}|$. By Claim 10, we know that the reduction algorithm returns a reduced instance $\mathcal{J}' = ((\Phi' = (\mathcal{V}, \widetilde{\mathcal{C}}), r, f, \mathcal{M} = (\mathcal{V}, I)), k, t')$ where the number of clauses is bounded by a polynomial function of the number of variables in the input formula. Specifically, we have that

$$|\widetilde{C}| \leq 2r \left(1 + \frac{1}{\varepsilon}\right) (d+1) |\mathcal{V}|^{d+1}.$$

Solution Lifting algorithm. The solution lifting algorithm Iden takes as input the instance $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$, the reduced instance $\mathcal{J}' = ((\Phi', r, f, \mathcal{M}), k, t')$ which is the output of $\mathcal{R}_C^{Mat}$ and a $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution to the instance $\mathcal{I}'$, for some $(\beta_0, \beta_1, \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$. Iden returns the approximate solution it receives as input.

Lemma 38. *For any $(\beta_0, \beta_1, \ldots, \beta_{r+1}) \in \mathbb{N}^{r+2}$, let Y denote any $(\beta_0, \beta_1, \ldots, \beta_{r+1})$-approximate solution for the instance $\mathcal{J}'$. Then, Y is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon), \beta_{r+1})$-approximate solution for the instance $\mathcal{I}$.*

Proof. Let S be an optimal solution to the original instance $\mathcal{J}$ and let $Y^\star$ denote an optimal solution for the reduced instance $\mathcal{J}'$. By Claim 12 and because $S \in I$,

being an optimal solution to $\mathcal{J}$, we know that S is also a feasible solution of size k for the reduced instance $\mathcal{J}'$. Thus, $|Y| \leq \beta_0|Y^\star| \leq \beta_0|S|$.

Since $Y^\star \in I$, and hence, $\ell^{r+1}(\mathcal{J}', k, Y^\star) = 0$. Thus, $\ell^{r+1}(\mathcal{J}', k, Y) \geq \frac{\ell^{r+1}(\mathcal{J}', k, Y^\star)}{\beta_{r+1}} = 0$, implying that $Y \in I$. Since both the input and the reduced instances are defined over the same matroid $\mathcal{M}$[3], we have that $\ell^{r+1}(\mathcal{J}, k, Y) = 0 = \frac{\ell^{r+1}(\mathcal{J}, k, S)}{\beta_{r+1}}$. By Lemma 30, we know that Y satisfies all the other constraint functions for $j \in [r]$.

Hence Y is also a $(\beta_0, \beta_1(1+\varepsilon), \ldots, \beta_r(1+\varepsilon), \beta_{r+1})$-approximate solution for the original instance $\mathcal{J}$. □

Hence, by Lemma 38, which proves that Iden is the desired solution lifting algorithm, and Claim 10, which proves that the number of clauses in the reduced instance is $2r(1+\frac{1}{\varepsilon})(d+1)|\mathcal{V}|^{d+1}$, we obtain the desired $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA.

12.3.4 Putting it all together: an approximate kernel for (M, F)-Max k-weight SAT

In this section, we prove Theorem 20.

Proof. Given an instance $\mathcal{J} = ((\Phi, r, f, \mathcal{M}), k, t)$, the reduction algorithm does the following.

- It first runs the reduction algorithm from $(1, 1+\frac{\varepsilon}{6}, \ldots, 1+\frac{\varepsilon}{6}, 1)$-APPA from Lemma 31.
- Then, it runs the reduction algorithm from $(1, 1+\frac{\varepsilon}{6}, \ldots, 1+\frac{\varepsilon}{6}, 1)$-APPA from Lemma 33.
- Finally it runs the the reduction algorithm from $(1, 1+\frac{\varepsilon}{6}, \ldots, 1+\frac{\varepsilon}{6}, 1)$-APPA from Lemma 37.

Thus, we get a $(1, (1+\frac{\varepsilon}{6})^3, \ldots, (1+\frac{\varepsilon}{6})^3, 1)$-APPA, or equivalently a $(1, 1+\varepsilon, \ldots, 1+\varepsilon, 1)$-APPA. This is because $\frac{1}{(1+\frac{\varepsilon}{6})^3} \geq \frac{1}{1+\varepsilon}$ for any $\varepsilon \in (0,1)$, and hence, for any $j \in [r]$, we have $\frac{t_j}{(1+\frac{\varepsilon}{6})^3} \geq \frac{t_j}{(1+\varepsilon)}$. Note that the first reduction algorithm only deletes clauses and hence if the input is a $K_{d,d}$-free formula, it returns a $K_{d,d}$-free formula as an output.

[3]Due to the hereditary property of the matroid, the subset of an independent set is also an independent set.

The first APPA returns a formula $\Phi' = (\mathcal{V}', \mathcal{C}')$ where the number of negative variables in $|\mathcal{V}'|$ is $\mathcal{O}\left(\frac{rk^2}{\varepsilon}\right)$. The second APPA bounds the number of positive variables in $\mathcal{V}'$ by $\left(\frac{k(k+|\mathcal{V}^{neg}|)}{\varepsilon}\right)^{\mathcal{O}(d^2 r)} = \left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^2 r)}$. Finally, the third APPA bounds the number of clauses and the size of the reduced instance by $|\mathcal{V}|^{\mathcal{O}(d)} = \left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^3 r)}$. Thus the size of the reduced instance gets bounded by $\left(\frac{rk}{\varepsilon}\right)^{\mathcal{O}(d^3 r)}$. All three APPAs do not violate the matroid constraint.

This completes the proof of Theorem 20.

□

Section V: Conclusion

Chapter 13

Conclusion and Future Directions

This thesis focuses on two canonical problems, CC-MaxSat and Maximum Coverage and investigates how structural restrictions on instances and relaxed notions of optimality can be leveraged to overcome well-established hardness barriers. The results presented in this work demonstrate that, although these problems are intractable under standard parameterizations and approximation guarantees, meaningful progress is possible when additional structure is present and when approximation and parameterization are treated as complementary tools rather than competing paradigms.

Beyond concrete algorithmic results, this thesis also contributes conceptually by unifying satisfiability and covering problems in the context of parameterized approximation. It also incorporates fairness and matroid constraints, and extends the theory of lossy kernelization to multi-criteria optimization settings. These developments highlight a broader theme: many hardness results that appear definitive under classical frameworks can be circumvented by carefully refining the notion of efficiency and solution quality. We believe that the techniques used here—particularly those based on sparse incidence structures, approximation-preserving reductions, and multi-criteria preprocessing—will find applications beyond the specific problems studied.

While the results presented in this thesis advance the understanding of parameterized approximation and lossy preprocessing for partial constraint satisfaction and covering problems, several limitations remain and point toward interesting directions for future work.

Our approximation-preserving reduction from CC-MaxSat (and its variants) to Maximum Coverage (and its variants) is fundamentally randomized. A natural

question is whether this reduction can be derandomized. A similar derandomization question also arises for our previously described algorithm for F-Maximum Coverage on bounded-frequency set systems. In that setting, it is particularly interesting to ask whether one can obtain an EPAS whose running time is single-exponential in k.

More broadly, the development of non-trivial PAS remains a compelling direction for future research. In particular, extending our results to F-Max k-weight SAT presents an intriguing and potentially challenging problem.

Bibliography

[1] Jan Arne Telle and Yngve Villanger. FPT algorithms for domination in sparse graphs and beyond. *Theor. Comput. Sci.*, 770:62–68, 2019.

[2] Alan Cobham. The intrinsic computational difficulty of functions. *In Proceedings of the 1964 Congress for Logic, methodology, and the Philosophy of Science*, 1964.

[3] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[4] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[5] Stephen A. Cook. *The Complexity of Theorem-Proving Procedures*, page 143–152. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023.

[6] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[8] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[9] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[10] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[11] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[12] Gérard Cornuéjols, George L. Nemhauser, and Laurence A. Wolsey. Worst-case and probabilistic analysis of algorithms for a location problem. *Oper. Res.*, 28(4):847–858, 1980.

[13] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 1998.

[14] Maxim Sviridenko. Best possible approximation algorithm for MAX SAT with cardinality constraint. *Algorithmica*, 2001.

[15] Markus Bläser and Bodo Manthey. Improved approximation algorithms for max-2sat with cardinality constraint. In Prosenjit Bose and Pat Morin, editors, *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, volume 2518 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 2002.

[16] Thomas Hofmeister. An approximation algorithm for MAX-2-SAT with cardinality constraint. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, volume 2832 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2003.

[17] Prasad Raghavendra and Ning Tan. Approximating csps with global cardinality constraints using SDP hierarchies. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 373–387. SIAM, 2012.

[18] Per Austrin and Aleksa Stankovic. Global cardinality constraints make approximating some max-2-csps harder. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 24:1–24:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[19] Pasin Manurangsi. A note on max k-vertex cover: Faster fpt-as, smaller approximate kernel and improved approximation. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms,*

SOSA 2019, January 8-9, 2019, San Diego, CA, USA, volume 69 of *OASIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[20] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[21] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 62–81. SIAM, 2020.

[22] Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.

[23] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Hooyeon Lee. Approximating low-dimensional coverage problems. In Tamal K. Dey and Sue Whitesides, editors, *Proceedings of the 28th ACM Symposium on Computational Geometry, Chapel Hill, NC, USA, June 17-20, 2012*, pages 161–170. ACM, 2012.

[24] Piotr Skowron and Piotr Faliszewski. Chamberlin-courant rule with approval ballots: Approximating the maxcover problem with bounded frequencies in FPT time. *J. Artif. Intell. Res.*, 60:687–716, 2017.

[25] Markus Bläser. Computing small partial coverings. *Inf. Process. Lett.*, 85(6):327–331, 2003.

[26] Pasin Manurangsi. A note on max k-vertex cover: Faster fpt-as, smaller approximate kernel and improved approximation. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[27] Daniel Lokshtanov, Fahad Panolan, and M. S. Ramanujan. Backdoor sets on nowhere dense SAT. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of

LIPIcs, pages 91:1–91:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[28] Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Trans. Algorithms*, 9(1):11:1–11:23, 2012.

[29] Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of vertex cover variants. *Theory Comput. Syst.*, 41(3):501–520, 2007.

[30] Chien-Chung Huang and François Sellier. Matroid-constrained vertex cover. *Theor. Comput. Sci.*, 965:113977, 2023.

[31] François Sellier. Parameterized matroid-constrained maximum coverage. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *ESA 2023*, volume 274 of *LIPIcs*, pages 94:1–94:16, 2023.

[32] Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Comb.*, 2(4):385–393, 1982.

[33] Gruia Cualinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

[34] Chandra Chekuri, Tanmay Inamdar, Kent Quanrud, Kasturi R. Varadarajan, and Zhao Zhang. Algorithms for covering multiple submodular constraints and applications. *J. Comb. Optim.*, 44(2):979–1010, 2022.

[35] Sayan Bandyapadhyay, Zachary Friggstad, and Ramin Mousavi. A parameterized approximation scheme for generalized partial vertex cover. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 93–105. Springer, 2023.

[36] Dániel Marx. A parameterized view on matroid optimization problems. In Hajo Broersma, Stefan S. Dantchev, Matthew Johnson, and Stefan Szeider, editors, *ACiD 2006*, volume 7 of *Texts in Algorithmics*, page 158. King's College, London, 2006.

[37] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual*

ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, pages 142–151. SIAM, 2014.

[38] Édouard Bonnet, Vangelis Th. Paschos, and Florian Sikora. Parameterized exact and approximation algorithms for maximum k-set cover and related satisfiability problems. *RAIRO Theor. Informatics Appl.*, 50(3):227–240, 2016.

[39] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014.

[40] Akanksha Agrawal, Pratibha Choudhary, Pallavi Jain, Lawqueen Kanesh, Vibha Sahlot, and Saket Saurabh. Hitting and covering partially. In *COCOON*, pages 751–763, 2018.

[41] Reuven Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *J. Algorithms*, 39(2):137–144, 2001.

[42] Nader H. Bshouty and Lynn Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998, Proceedings*, volume 1373 of *Lecture Notes in Computer Science*, pages 298–308. Springer, 1998.

[43] Dorit S. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In Klaus Jansen and Dorit S. Hochbaum, editors, *Approximation Algorithms for Combinatorial Optimization, International Workshop APPROX'98, Aalborg, Denmark, July 18-19, 1998, Proceedings*, volume 1444 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 1998.

[44] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[45] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, page 25. IEEE Computer Society, 2002.

[46] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM*

SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018, pages 1283–1296. ACM, 2018.

[47] Mitali Bafna, Karthik C. S., and Dor Minzer. Near optimal constant inapproximability under ETH for fundamental problems in parameterized complexity. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 2118–2129. ACM, 2025.

[48] Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. On equivalence of parameterized inapproximability of k-median, k-max-coverage, and 2-csp. In Édouard Bonnet and Pawel Rzazewski, editors, *19th International Symposium on Parameterized and Exact Computation, IPEC 2024, September 4-6, 2024, Royal Holloway, University of London, Egham, United Kingdom*, volume 321 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[49] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019.

[50] Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1546–1558. SIAM, 2017.

[51] Aditya Anand, Euiwoong Lee, Jason Li, and Thatchaphol Saranurak. Approximating small sparse cuts. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 319–330. ACM, 2024.

[52] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Almost optimal time lower bound for approximating parameterized clique, csp, and more, under ETH. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 2136–2144. ACM, 2025.

[53] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized inapproximability hypothesis under exponential time hypothesis. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings*

of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024, pages 24–35. ACM, 2024.

[54] Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018.

[55] Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 375–384. ACM, 2006.

[56] Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Fpt-approximation for FPT problems. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 199–218, 2021.

[57] Daniel Lokshtanov, Abhishek Sahu, Saket Saurabh, Vaishali Surianarayanan, and Jie Xue. Parameterized approximation for capacitated d-hitting set with hard capacities. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 1565–1592. SIAM, 2025.

[58] Pasin Manurangsi. Improved FPT approximation scheme and approximate kernel for biclique-free max k-weight SAT: greedy strikes back. *Theor. Comput. Sci.*, 1028:115033, 2025.

[59] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. *SIAM J. Comput.*, 53(6):S20–205, 2024.

[60] Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 528–541. ACM, 2023.

[61] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 184–192. IEEE, 2021.

[62] Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In Hamed Hatami, Pierre McKenzie, and Valerie King,

editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237. ACM, 2017.

[63] R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Many birds with one stone: multi-objective approximation algorithms. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 438–447. ACM, 1993.

[64] Fabrizio Grandoni, R. Ravi, Mohit Singh, and Rico Zenklusen. New approaches to multi-objective optimization. *Math. Program.*, 146(1-2):525–554, 2014.

[65] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 182–191. IEEE Computer Society, 1995.

[66] Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. In *Encyclopedia of Algorithms*, pages 335–338. 2016.

[67] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020.

[68] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[69] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 922–934. Springer, 2015.

[70] C. Hyltén-Cavallius. On a combinatorical problem. *Colloquium Mathematicae*, 6(1):61–65, 1958.

[71] P Kővári, Vera T Sós, and Pál Turán. On a problem of zarankiewicz. In *Colloquium Mathematicum*, volume 3, pages 50–57. Polska Akademia Nauk, 1954.

[72] Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of generalized vertex cover problems. In Frank K. H. A. Dehne, Alejandro López-Ortiz, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 36–48. Springer, 2005.