PARAMETERIZING FROM THE EXTREMES: FEASIBLE PARAMETERIZATIONS OF SOME NP-OPTIMIZATION PROBLEMS

Somnath Sikdar

The Institute of Mathematical Sciences, Chennai.

A thesis submitted to the

Board of Studies in Mathematical Sciences

in partial fulfilment of the requirements

for the Degree of

Doctor of Philosophy

of

Homi Bhabha National Institute

June 2010

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we recommend that the dissertation prepared by Somnath Sikdar titled *Parameterizing From the Extremes: Feasible Parameterizations of Some NP-Optimization Problems* be accepted as fulfilling the requirements for the Degree of Doctor of Philosophy.

---------- *Date:*
Chair: V. Arvind

---------- *Date:*
Convener: Venkatesh Raman

---------- *Date:*
Member 1: Sunil Chandran

---------- *Date:*
Member 2: Meena Mahajan

---------- *Date:*
Member 3: K.V. Subrahmanyam

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

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

---------- *Date:*

Advisor : Venkatesh Raman

Declaration

I declare that the thesis titled *Parameterizing From the Extremes: Feasible Parameterizations of Some NP-Optimization Problems* is a record of the work carried out by me during the period August 2004 to July 2009 under the supervision of Prof. Venkatesh Raman. This work is original and it has not been submitted earlier as a whole or in part for a degree, diploma, associateship or fellowship at this or any other institute or university.

Chennai,
June 2010.

Somnath Sikdar

Certificate

I certify that the thesis titled *Parameterizing From the Extremes: Feasible Parameterizations of Some NP-Optimization Problems* submitted for the degree of Doctor of Philosophy by Somnath Sikdar is a record of the research carried out by him during the period August 2004 to July 2009 under my supervision. This work has not formed the basis for the award of any degree, diploma, associateship or fellowship at this or any other institute or university.

Chennai,
June 2010.

Venkatesh Raman

To my family.

I long to accomplish a great and noble task, but it is my chief duty to accomplish small tasks as if they were great and noble.

Helen Keller (1880-1968).

Abstract

Parameterized complexity is a newly developed sub-area of computational complexity that allows for a more refined analysis of problems that are considered hard in the classical sense. In contrast to the classical theory where the complexity of a problem is measured in terms of the input size only, parameterized complexity seeks to exploit the internal structure of a problem. The complexity of a problem in this case is measured not just in terms of the input size but in terms of the input size and, what is called, the *parameter*.

A parameterized problem is a decision problem whose instances consist of tuples (I, k), where $n = |I|$ is the size of the input instance and k is the parameter. The goal here is to design algorithms that decide whether (I, k) is a YES-instance in time $f(k) \cdot n^{O(1)}$, where f is a computable function of k alone, as against a trivial algorithm with running time $n^{k+O(1)}$. Problems that admit such algorithms are said to be *fixed-parameter tractable* and FPT denotes the class of all fixed-parameter tractable problems. The parameter, however, is not unique and often there are several ways in which a problem can be parameterized. This is, in fact, one of the strengths of parameterized complexity as it allows the same problem to be analyzed in different ways depending on the parameter. In this thesis, we study different parameterizations of NP-optimization problems with the intent of identifying those parameterizations that are feasible and most likely to be useful in practice.

A commonly studied parameterization of NP-optimization problems is the *standard parameterized version*, where the parameter is the solution size. We begin by showing that a number of NP-optimization problems, and in particular problems in MAX SNP, have the property that their optimum solution size is bounded below by an unbounded function of the input size. We show that the standard parameterized version of these problems is trivially in FPT and we argue that the natural parameter in such cases is the deficit between the optimum and the lower bound. That is, one ought to parameterize *above* the guaranteed lower bound and we call such a parameterization an "above-guarantee" parameterization. One can similarly define parameterizations below a guaranteed upper bound. We then introduce the notion of "tight" lower and upper bounds and exhibit problems for which the above-guarantee and below-guarantee parameterization with

respect to a tight bound is fixed-parameter tractable or W-hard. We show that if we parameterize "sufficiently" above or below tight bounds, then these parameterized versions are not in FPT, unless P = NP, for a class of NP-optimization problems.

We then consider related questions in the approximation algorithms setting. We investigate the possibility of obtaining an approximation algorithm for an NP-optimization problem that is an ϵ-fraction better than the best-known approximation ratio for the problem. Since the best-known ratio could also be the approximation lower-bound for the problem, the algorithm in question could possibly have a worst-case exponential-time complexity. But the challenge is to obtain moderately exponential-time algorithms, whose run-time is possibly a function of ϵ and the input-size, that deliver $(\alpha + \epsilon)$-approximate solutions. We discuss a technique that allows us to obtain such algorithms for a class of NP-optimization problems.

We next study the parameterized complexity (and occasionally the approximability) of a number of concrete problems: Kőnig Subgraph problems, Unique Coverage and its weighted variant, a version of the Induced Subgraph problem in directed graphs, and the Directed Full-Degree Spanning Tree problem. The Kőnig Subgraph problem is actually a set of problems where the goal is to decide whether a given graph has a Kőnig subgraph of a certain size. A graph is Kőnig if the size of a maximum matching equals that of a minimum vertex cover in the graph. Such graphs have been studied extensively from a structural point-of-view. In this thesis, we initiate the study of the parameterized complexity and approximability of finding Kőnig subgraphs of a given graph. We will see that one of the Kőnig Subgraph problems, namely Kőnig Vertex Deletion, is closely related to a well-known problem in parameterized complexity called Above Guarantee Vertex Cover. While studying the parameterized complexity of Kőnig Vertex Deletion, we will also see some interesting structural relations between matchings and vertex covers of a graph.

Unique Coverage is a natural maximization version of the well-known Set Cover problem and has applications in wireless networking and radio broadcasting. It is also a natural generalization of the well-known Max Cut problem. In this problem we are given a family of subsets of a finite universe and a nonnegative integer k as parameter, and the goal is to decide whether there exists a subfamily that covers at least k elements exactly once. We show that this problem is fixed-parameter tractable by exhibiting a problem kernel with 4^k sets. We also consider a weighted variant of it called Budgeted Unique Coverage and, by an application of the color-coding technique, show it to be fixed-parameter tractable.

Our application of color-coding uses an interesting variation of k-perfect hash families where for every s-element subset S of the universe, and for every k-element subset X of S, there exists a function that maps X injectively and maps the remaining elements of S into a different range. Such

families are called (k, s)-hash families and were studied before in the context of coding theory. We prove, using the probabilistic method, the existence of such hash families of size smaller than that of the best-known s-perfect hash families. Explicit constructions of such hash families of size promised by the probabilistic method is open.

We study a version of the Induced Subgraph problem in directed graphs defined as follows: given a hereditary property $\mathcal{P}$ on digraphs, an input digraph D and a nonnegative integer k, decide whether D has an induced subdigraph on k vertices with property $\mathcal{P}$. We completely characterize hereditary properties for which this induced subgraph problem is W[1]-complete for two classes of directed graphs: general directed graphs and oriented graphs. We also characterize those properties for which the induced subgraph problem is W[1]-complete for general directed graphs but fixed-parameter tractable for oriented graphs.

We also study a directed analog of a problem called Full Degree Spanning Tree which has applications in water distribution networks. This problem is defined as follows: given a digraph D and a nonnegative integer k, decide whether there exists a spanning out-tree of D with at least k vertices of full out-degree. We show that this problem is W[1]-hard on two important digraph classes: directed acyclic digraphs and strongly connected digraphs. In the dual version, called Reduced Degree Spanning Tree, one has to decide whether there exists a spanning out-tree with at most k vertices of reduced out-degree. We show that this problem is fixed-parameter tractable and admits a problem kernel with at most $8k$ vertices on strongly connected digraphs and $O(k^2)$ vertices on general digraphs. We also give an algorithm for this problem on general digraphs with run-time $O^*(5.942^k)$.

Acknowledgements

I've had the good fortune of meeting a number of wonderful people who have, over the years, shaped my ideas and helped me in my research career. I take this opportunity to thank them.

Firstly, I'm indebted to Venkatesh Raman for his guidance, support and encouragement over the years. He introduced me to the area of parameterized complexity and was extraordinarily patient in explaining his ideas to me. He was more of a friend to me than a supervisor and this thesis would certainly not have been possible without his support. I thank Meena Mahajan for spending so many hours in discussions during my initial years as a PhD student. She was very patient and helped boost my confidence. I also thank my doctoral committee members—V. Arvind, Meena Mahajan and K.V. Subrahmanyam for their helpful advice.

Part of the work for this thesis was supported by DST-DAAD project no. INT/DAAD/P-138/2006 titled *Provably Efficient Algorithms for Computationally Hard Problems* between Venkatesh Raman and Rolf Niedermeier. This included two academic visits to the Friedrich-Schiller University, Jena, Germany (August 2006–October 2006 and June 2007–July 2007). I thank Rolf Niedermeier and all the students in the theoretical computer science group at FSU, Jena, and in particular, Michael Dom and Hannes Moser, for their hospitality and for making both my stays at Jena such a pleasant experience. I also thank them for several fruitful discussions we'd had.

I'm grateful to V. Arvind, Kamal Lodaya, Meena Mahajan, Venkatesh Raman, R. Ramanujam, and C.R. Subramanian for their valuable teaching which helped me during my formative years. I thank all my teachers at the Indian Statistical Institute, Kolkata for instilling in me an interest in theoretical computer science. I thank Mike Fellows for some very inspiring lectures on parameterized complexity.

I thank all my coauthors for letting me to use results obtained with them as part of this thesis. I especially thank Saket Saurabh for spending so much time discussing with me and for some very enjoyable dinners at Jena. I thank Daniel Lokshtanov, Sounaka Mishra, Neeldhara Misra, Rahul Muthu, N. Narayanan, Geevarghese Philip, and Srikanth Srinivasan for taking time out to discuss with me. I thank my batchmates Bireswar Das, Nutan Limaye, Sunil Simon, and S. Sheerazuddin for all the discussions we've had during

the period of our course-work. Some of the people listed here were also part of the Graph Theory Group which met once a week to discuss problems from Douglas West's *Introduction to Graph Theory*. I thank N.R. Aravind, Nutan Limaye, Rahul Muthu, N. Narayanan, Saket Saurabh for being such an enthusiastic part of this group.

During these past years, I was in enjoyable company of friends with whom I dined frequently. Frequent too were the late-night parties! I thank all the people who were involved in organizing these parties and for making them so memorable. I thank Bireswar Das, Pushkar Joglekar, Saptarshi Mandal, Partha Mukhopadhyay, Rahul Muthu, Prajakta Nimbhorkar, among others for making my stay at IMSc a pleasant one. I thank the administrative staff at IMSc for the all the help they provided me during these past years.

Finally, I thank my parents who encouraged me to pursue a career in research and for their love and support. I thank Satarupa, Mithu, Vinay and little Aryaman for their love, understanding and encouragement. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Parameterized complexity theory is a new branch of complexity theory developed by Downey and Fellows in a series of papers in the early 1990s. This theory provides a framework for a more refined analysis of algorithmic problems that are considered hard in the classical sense.

In classical complexity theory, one categorizes problems based on the amount of a resource, usually time or space, required to solve them. But how does one measure the amount of a resource? This was answered by Hartmanis and Stearns who introduced the notion of *input size* and the idea of measuring the amount of a resource as a function of the input size [79]. This led to the subsequent development of classical complexity theory as we know it today with a relatively clean theory of intractability. But if we measure the complexity of a problem solely in terms of its input size, then we necessarily ignore all structural information about the input instance in the resulting theory. This is hardly a desirable situation as the absence of structural information may make a problem appear harder than what it actually is. Parameterized complexity theory measures the complexity of a problem not from the standpoint of the input size alone, but in terms of both the input size and a parameter. In many situations, the parameter is a numerical value that captures some structural information about the input instance. But there are no restrictions on what the parameter can be and there are cases when the parameter is a structural component of the input, for example, a graph.

The fundamental notion in parameterized complexity is that of *fixed-parameter tractability*. It extends the classical notion of a feasible algorithm as one whose running time is polynomial in the input size to admit algorithms whose running time is polynomial in the input size but is possibly non-polynomial in the parameter. An important contribution of the theory is the development of a framework where one can show that certain problems are *not* fixed-parameter tractable. In this framework, one classifies problems into complexity classes by means of suitable reductions. Since the theory is two-dimensional, the reductions and the resulting complexity classes are

more involved than in the classical case.

In this chapter we introduce the basic definitions in parameterized complexity. We discuss various ways in which parameters can arise and why certain parameterizations are not useful. We also provide a very short survey of the complexity classes that crop up in parameterized complexity theory. For a general introduction to this area, one can refer to [47, 59, 107].

1.1 Basic Definitions

We start by defining several key terms used in parameterized complexity. Let Σ be a nonempty finite alphabet and let Σ^* denote the set of all finite strings over Σ. A *parameterized problem* Q is a subset of $\Sigma \times \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of nonnegative integers. An instance of a parameterized problem is therefore a pair (I, k), where k is the parameter. Examples of parameterized problems include:

Example 1.1. Vertex Cover

Input:	An undirected graph $G = (V, E)$ and a nonnegative integer k.
Parameter:	The integer k.
Question:	Does there exist a vertex subset S of size at most k such that every edge of G has an end-point in S?

Example 1.2. Dominating Set

Input:	An undirected graph $G = (V, E)$ and a nonnegative integer k.
Parameter:	The integer k.
Question:	Does there exist a vertex subset S of size at most k such that every vertex of G has a neighbor in S?

Example 1.3. Max Sat

Input:	A boolean formula ϕ in conjunctive normal form with n variables and m clauses; a nonnegative integer k.
Parameter:	The integer k.
Question:	Does there exist an assignment that satisfies all but k clauses of ϕ?

In the framework of parameterized complexity, the running time of an algorithm is viewed as a function of two quantities: the size of the problem instance *and* the parameter.

Definition 1.1. A parameterized problem Q is said to be *fixed-parameter tractable* if there exists an algorithm that takes as input an instance (I, k) of Q and decides whether it is a YES- or a NO-instance in time $O(f(k) \cdot p(n))$, where $n = |I|$, f is some computable function and p some polynomial. Such an algorithm is sometimes called an *FPT-algorithm for Q*.

The class FPT consists of all fixed-parameter tractable problems. We assume that an FPT-algorithm for a parameterized problem produces a witness whenever it outputs YES.

1.1.1 Kernels

Closely related to the notion of an FPT-algorithm is the concept of a kernel.

Definition 1.2. Let Q be a parameterized problem. A *kernelization of Q* is a polynomial-time many-one reduction from Q to Q that maps a given instance (I, k) to an equivalent instance (I', k') such that $|I'| \leq g(k)$ and $k' \leq h(k)$, where g and h are two computable functions. The two instances are equivalent in the sense that (I, k) is a YES-instance if and only if (I', k') is a YES-instance.

The function g is called the *size of the kernel* and the polynomial-time many-one reduction is a *kernelization algorithm* for Q. A parameterized problem is *kernelizable* if it admits a kernelization algorithm.

One reason why the study of kernels is important is because of

Proposition 1.1 ([59])**.** *Let Q be a parameterized problem that is decidable. Then Q is kernelizable if and only if it is fixed-parameter tractable.*

Proof. Suppose Q is kernelizable and let $\mathcal{A}$ be a kernelization algorithm for it. Then the following algorithm is an FPT-algorithm for Q: given an instance (I, k) of Q, it first computes $\mathcal{A}(I, k) = (I', k')$ and then uses the decision algorithm for Q to decide whether $(I', k') \in Q$. Since $|I'| \leq h(k)$, the running time of the decision algorithm is bounded in terms of the parameter k.

Conversely suppose that $\mathcal{A}$ is an FPT-algorithm for Q with running time $f(k) \cdot |I|^c$, for some computable f and some $c \in \mathbb{N}$. The following algorithm $\mathcal{A}'$ computes a kernel for Q. Given an instance (I, k), the algorithm $\mathcal{A}'$ simulates the FPT-algorithm $\mathcal{A}$ for $|I|^{c+1}$ steps. If $f(k) \leq |I|$, then $\mathcal{A}$ stops and accepts or rejects within these many steps. In this case, the algorithm $\mathcal{A}'$ stops and outputs a trivial YES- or NO-instance, respectively. If $\mathcal{A}$ does not stop in $|I|^{c+1}$ steps then it must be that $|I| \leq f(k)$, and $\mathcal{A}'$ outputs (I, k). Therefore if (I', k') is the instance output by $\mathcal{A}'$ then $|I'| \leq f(k) + O(1)$. □

As the proof of Proposition 1.1 shows, although the notions of kernelizability and fixed-parameter tractability are equivalent, a good FPT-algorithm for a parameterized problem does not necessarily yield a kernel

of small size. In fact, if the parameterized problem is NP-complete, then any FPT-algorithm for it will run in time superpolynomial (or worse) in the parameter, assuming P $\neq$ NP. The kernel obtained from such an algorithm is therefore at least superpolynomial in the parameter and so it is interesting to ask if the kernel size can be made smaller—in particular, whether it can be made polynomial in the parameter. It is therefore of independent interest to consider kernelization algorithms for parameterized problems.

A common technique to establish a problem-kernel is to devise a set of *reduction rules* which when applied to the input instance (in some specified sequence) produces the kernel. Reduction rules may be thought of as steps in the kernelization algorithm. Formally,

Definition 1.3. A *reduction rule* for a parameterized problem Q is an algorithm that takes an instance (I, k) of Q and, in time polynomial in $|I|$ and k, outputs either

1. YES or NO, in which case the input instance is a YES- or NO-instance, respectively, or
2. an "equivalent" instance (I', k') of Q such that $k' \leq k$.

Two instances (I, k) and (I', k') are *equivalent* if the following condition holds: (I, k) is a YES-instance if and only if (I', k') is a YES-instance.

Definition 1.4. An instance (I, k) of a parameterized problem Q is *reduced with respect to (w.r.t.) a set $\mathcal{R}$ of reduction rules* if the instance (D', k') output by any reduction rule in $\mathcal{R}$ is the original instance (D, k) itself.

As an example of how to use reduction rules to obtain a problem kernel we consider the VERTEX COVER problem. Recall that an instance of this problem consists of an undirected graph $G = (V, E)$ and a nonnegative integer k and the question is to decide whether G has a vertex cover of size at most k. Observe that any vertex of G with degree at least $k + 1$ must be in every size k vertex cover of G. Our reduction rules, in this case, take the following form:

Rule 0. Remove vertices of degree zero from the graph.

Rule 1. If $u \in V(G)$ is of degree at least $k + 1$, set $S \leftarrow S \cup \{u\}$, $k \leftarrow k - 1$ and $G \leftarrow G - u$. Return (G, k).

We assume that S is the vertex cover that is being constructed. Here is a kernelization algorithm for VERTEX COVER.

1. Reduce the input instance (G, k) w.r.t. Rules 0 and 1 to obtain (G', k'). If $k' < 0$ then G does not have a size k vertex cover and reject.
2. If G' has more than $k'(k + 1)$ vertices, reject. Else return (G', k').

The bound $k'(k+1)$ in Step 2 is justified by the fact that a simple undirected graph with a k'-vertex cover and degrees bounded above by k and below by 1 has no more than $k'(k+1)$ vertices. This algorithm, in polynomial time, produces an equivalent instance which has at most $k^2 + k$ vertices. Buss and Goldsmith [24] attribute this algorithm to S. Buss. Subsequently Chen et al. [30] observed that using an analysis of a linear program for VERTEX COVER due to Nemhauser and Trotter [106], one can obtain a kernel with at most $2k$ vertices.

Proving polynomial bounds on the size of the kernel for different parameterized problems has been an important practical aspect in the study of the parameterized complexity of NP-hard problems, and many positive results are known. See [71] for a survey of kernelization results. Recently Bodlaender et al. [18] building on the work of Fortnow and Santhanam [63] developed a lower-bound technique that allows one to prove that a number of parameterized problems do not admit polynomial kernels unless PH (the polynomial hierarchy) collapses to the third level. Dom et al. [46] have recently extended the techniques to show many more parameterized problems do not admit polynomial kernels under the same complexity-theoretic assumptions.

1.2 Types of Parameterizations

We mentioned that the parameter is a numeric value that usually captures some structural information about the input instance. But there are myriad ways in which numbers can arise naturally in problem specifications and therefore the parameter need not be unique. It is often the case that a problem admits several parameterizations and the complexity of these variants are different from one another. A good example of such a problem is the LONGEST COMMON SUBSEQUENCE problem which is defined below.

Input:	A set R of strings $r_1, \ldots, r_l$ over a finite alphabet Σ and a nonnegative integer k.
Question:	Does there exist a string $s \in \Sigma^*$ of length at least k such that s is a subsequence of each string in R?

A string s is a subsequence of r if it can be obtained from r by deleting some characters from r. LONGEST COMMON SUBSEQUENCE is NP-complete and the following parameterized variants of it have been studied in the literature (see [110] for more details).

1. Parameter: l.
2. Parameter: k.

Parameter	*Alphabet size* $\|\Sigma\|$ *unbounded*	*parameter*	*constant*
l	W[t]-hard ($t \geq 1$)	W[t]-hard ($t \geq 1$)	W[1]-hard
k	W[2]-hard	FPT	FPT
l, k	W[1]-complete	FPT	FPT

Table 1.1: Complexity of parameterized variants of Longest Common Subsequence.

3. Parameter: l, k.

4. Parameter: $k, |\Sigma|$.

The parameterized complexity of these variants are summarized in Figure 1.1. The hardness results in the table indicate that the corresponding parameterized variants are unlikely to be in FPT.

Although one can parameterize an optimization problem in several ways, it is often the case that the parameter chosen is the solution size and this particular variant is then called the standard parameterized version. For an NP-optimization problem Q, the *standard parameterized version* Q_{par} is the following decision problem.

Input: A tuple (I, k), where I is an instance of Q and k is a non-negative integer, the parameter.

Question: Is the optimum solution size of I at least k (if Q is a maximization problem) or at most k (if Q is a minimization problem)?

Although the standard parameterized version is the most popularly studied parameterization of NP-optimization problems, we will show in the next chapter that this version may not always be interesting. In particular, there are NP-optimization problems for which the standard version is trivially fixed-parameter tractable.

Consider the (standard parameterization) of Vertex Cover: given a graph $G = (V, E)$ on n vertices, decide whether G has a vertex cover of size at most k. The best-known algorithm for Vertex Cover is due to Chen et al. [30] and runs in time $O(1.2852^k + kn)$. Observe that if G has a matching of size μ then any vertex cover of G picks at least one vertex from each edge of the matching, and hence is of size at least μ. Therefore if G has a perfect matching, then for G to have a size k vertex cover we must have $k \geq n/2$. The point being made here is that for a large class of graphs $k = \Omega(n)$, and therefore *any* algorithm for the standard version either takes time exponential in n (if $k \geq n/2$) or is trivial (if $k < n/2$).

Now consider an alternative parameterization: does G have a vertex cover of size at most $\mu + k$? Note that the parameter k here is the size of the vertex cover above the matching size. Since the matching size is a guaranteed lower bound on the vertex cover size, we call this version Above Guarantee Vertex Cover. In this thesis we show that this version admits an algorithm with running time $O(15^k \cdot k \cdot m^3)$, where m is the number of edges in the input graph. Note that on the class of graphs with a perfect matching, this running time is better than any running time of the form $O(c^{k'} \cdot \text{poly}(n))$, where $k' = \mu + k$ is the standard parameter.

Just as one can parameterize above guaranteed values, one can consider parameterizations below guaranteed values. As an example, consider the following variant of Vertex Cover: given a planar graph $G = (V, E)$ on n vertices and an integer parameter k, does G admit a vertex cover of size $\lfloor 3n/4 \rfloor - k$? To see why this is a parameterization below a guaranteed value, note that any planar graph can be colored using four colors and hence has an independent set of size at least $\lceil n/4 \rceil$, and therefore a vertex cover of size at most $\lfloor 3n/4 \rfloor$. Thus $\lfloor 3n/4 \rfloor$ vertices suffice to cover the edges of a planar graph and this is indeed a parameterization below a guaranteed upper bound. Also note that a planar graph has a vertex cover of size $\lfloor 3n/4 \rfloor - k$ if and only if it has an independent set of size at least $\lceil n/4 \rceil + k$. The parameterized complexity of this problem is still open.

In short, parameters can arise in numerous ways and one can conceive of several different ways to parameterize a given problem. We discuss these issues in detail in the next chapter and this is in fact one the major themes of this thesis. For the time being, we turn to a brief discussion on parameterized intractability.

1.3 Reductions and Parameterized Intractability

Parameterized complexity theory not only provides a rich algorithmic toolkit to show positive results, but it also provides a framework that allows us to show that certain problems are unlikely to be fixed-parameter tractable. As with classical NP-completeness theory, the starting point in the development of any such framework is a suitable notion of a reduction in the parameterized setting. A reduction in the parameterized setting is defined as follows:

Definition 1.5. Let L_1 and L_2 be parameterized languages. We say that L_1 *fixed-parameter reduces* to L_2, denoted by $L_1 \leq_{\text{FPT}} L_2$, if there exist functions $f, g \colon \mathbb{N}_0 \to \mathbb{N}_0$, $\Phi \colon \Sigma^* \times \mathbb{N}_0 \to \Sigma^*$ and a polynomial $p(\cdot)$ such that for any instance (I, k) of L_1,

1. $(\Phi(I, k), g(k))$ is an instance of L_2 computable in time $f(k) \cdot p(|I|)$;

2. (I, k) is a YES-instance of L_1 if and only if $(\Phi(I, k), g(k))$ is a YES-instance of L_2.

We say that L_1 and L_2 are *fixed-parameter equivalent* if and only if $L_1 \leq_{\text{FPT}} L_2$ and $L_2 \leq_{\text{FPT}} L_1$.

A fixed-parameter reduction (briefly, an fp-reduction) differs from the familiar many-one Karp-reduction in two ways. Firstly, the time taken could be exponential in the parameter k in contrast to a Karp-reduction where the time taken can be explicitly specified by a polynomial function. Secondly, the "new" parameter must be a function of the old parameter only, that is, it should have no dependence on the input size $|I|$. In a Karp-reduction it is possible that the new parameter is a function of both k and $|I|$. It is also easy to see that if $L_1 \leq_{\text{FPT}} L_2$ and $L_2 \in \text{FPT}$ then $L_1 \in \text{FPT}$. That is, the class FPT is downward-closed under fp-reductions. It is also easy to verify that an fp-reduction is reflexive and transitive.

If we parallel NP-completeness theory then the other ingredient, besides the notion of a reduction, is the identification of a class of problems which is unlikely to be fixed-parameter tractable. In parameterized complexity theory, this class of problems is defined using a set of satisfiability problems on Boolean circuits. To describe this class of problems, we first need some definitions.

A Boolean circuit C with n input nodes is a directed acyclic graph with n nodes of in-degree zero called *input gates*, with each input gate labelled by either a positive literal x_i or a negative literal $\bar{x}_i$, where $1 \leq i \leq n$. The remaining nodes of the circuit are called *gates* and are labelled by a Boolean operator: either AND or OR. The circuit has a designated gate of out-degree zero called the *output gate*. A circuit C with n input nodes computes an n-ary Boolean function in a natural way. We say that $x \in \{0, 1\}^n$ satisfies C if the value computed by C on input x is 1. The *weight* of x is the number of 1s in x. A gate is called *large* if its fan-in exceeds some pre-agreed bound > 2; otherwise it is called *small*. The *weft* of a circuit is the maximum number of large gates on any path from the input gates to the output gate. The *depth* of a circuit is a maximum number of gates on any path from the input gates to the output gate. The *size* of a circuit C, denoted by $|C|$, is the total number of nodes and arcs in it.

We are now ready to define our class of circuit satisfiability problems:

WEIGHTED WEFT t DEPTH h CIRCUIT SATISFIABILITY (WCS(t, h))

Input:	A weft t depth h decision circuit C with n input nodes such that $\|C\| = \text{poly}(n)$.
Parameter:	A nonnegative integer k.
Question:	Does C have a satisfying assignment of weight exactly k?

The pre-agreed bound on the fan-in of small gates is some arbitrary constant. The theory that we describe does not depend on what this bound actually is. Note that a Boolean 3-Cnf formula (one whose clauses has at most three literals) can be expressed as a circuit of weft one and depth two. Therefore Weighted 3-Cnf Satisfiability, which is the problem of deciding whether a given 3-Cnf formula has a satisfying assignment of Hamming weight exactly k, is subsumed by WCS$(1, 2)$. Define $L_{C(t,h)}$ to be the set of tuples (C, k), such that C is a weft t depth h decision circuit that admits a weight k satisfying assignment. The basic hardness classes of parameterized complexity theory are defined as follows.

Definition 1.6. A parameterized language L is in the class W$[t]$ if and only if L fp-reduces to $L_{C(t,h)}$ for some constant h.

We first note that FPT $\subseteq$ W[1]. To see this, let L be a parameterized language in FPT and let (I, k) be an instance of L. One can decide whether (I, k) is a yes- or no-instance in time $f(k) \cdot |I|^c$, for some computable f and $c \in \mathbb{N}$, and accordingly produce a yes- or no-instance, respectively, of Weighted 3-Cnf Satisfiability. This shows that L fp-reduces to $L_{C(1,2)}$, and consequently FPT $\subseteq$ W[1]. Since the WCS(t, h) problem is subsumed by the WCS$(t + 1, h)$ problem, we have W$[t] \subseteq$ W$[t + 1]$ for all $t \geq 1$.

Finally we define:

Weighted Circuit Satisfiability

Input:	A Boolean decision circuit C with n input nodes such that $\|C\| = \text{poly}(n)$.
Parameter:	A nonnegative integer k.
Question:	Does C have a satisfying assignment of weight exactly k?

and W$[P]$ to be the class of parameterized languages that fp-reduce to Weighted Circuit Satisfiability. Since the set of all polynomial-sized circuits include all polynomial-sized weft t depth h circuits for all t and h, we have W$[t] \subseteq$ W$[P]$ for all $t \geq 1$. We therefore obtain an infinite hierarchy of complexity classes

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{W}[t] \subseteq \cdots \subseteq \text{W}[P]$$

called the *W-hierarchy* (W for weft). It is conjectured that these containments are proper.

One can now define the usual notions of hardness and completeness. A parameterized language L is *hard for a class* W$[t]$, if all languages in W$[t]$ fp-reduce to L. The language L is W$[t]$-complete, if $L \in$ W$[t]$ and L is hard for W$[t]$. Note that, by definition, the language $L_{C(t,h)}$ is complete for the class W$[t]$ for $t \geq 1$.

To understand why one suspects the class W[1] to be the analog of the class NP, consider the following problem:

Short Turing Machine Acceptance

Input:	A nondeterministic Turing machine M, a string x and an integer k
Parameter:	The integer k.
Question:	Does M have a computation path accepting x in at most k steps?

This is a parameterized variant of the NP-complete Turing Machine Acceptance problem. To quote Downey and Fellows [47]: "a nondeterministic Turing machine is such an opaque and generic object that it simply does not seem reasonable that we should be able to decide in polynomial time whether a given Turing machine on a given input has some accepting path. It seems to us that if one accepts the philosophical argument that Turing Machine Acceptance is intractable, the same reasoning would suggest that Short Turing Machine Acceptance is *fixed-parameter* intractable".

The belief that Short Turing Machine Acceptance is the key fixed-parameter intractable problem and the fact that Short Turing Machine Acceptance is W[1]-complete lend credence to the belief that W[1] is indeed the basic intractability class in the parameterized world. There are numerous other problems known to be complete for W[1]. We list some of them in the next theorem. For the definitions of these problems see [47, 19].

Theorem 1.1. [47] *The following problems are complete for the class* W[1].

1. Short Turing Machine Acceptance.
2. Independent Set.
3. Clique.
4. Weighted c-Cnf Satisfiability, *for a fixed* $c \geq 1$.

There are several problems complete for W[2].

Theorem 1.2. [47] *The following problems are complete for the class* W[2].

1. Dominating Set.
2. Hitting Set.
3. Set Cover.

Finally, it is interesting to note that there are several natural problems that are hard for every level of the W-hierarchy.

Theorem 1.3. [19] *The following problems are* W[t]*-hard for all* $t \geq 1$.

1. Longest Common Subsequence *parameterized by the number of strings.*
2. Bandwidth.
3. Perfect Phylogeny.
4. DNA Physical Mapping.

1.4 Notation Used in this Thesis

In this section we fix notation that we use throughout this thesis. Notation specific to a particular chapter will be described in that chapter itself.

We use two notations to describe the running time of an algorithm. The first notation we use is the familiar "big-oh" notation. Let $f, g\colon \mathbb{N} \to \mathbb{N}$ be functions on natural numbers. We write $f(n) = O(g(n))$ if there exist constants $c, n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $f(n) \leq c \cdot g(n)$. The other notation is the O^* notation and is primarily used for exponential-time algorithms. If $h\colon \mathbb{N} \to \mathbb{N}$ is a super-polynomial function on natural numbers, then $O^*(h(n)) = O(h(n) \cdot p(n))$, where $p(\cdot)$ is some polynomial function. That is, the $O^*(\cdot)$ notation suppresses polynomial terms in the running time expression. From time to time, we will also use the $O^*(\cdot)$ notation to describe the running time of a parameterized algorithm. Thus we write $O^*(f(k))$ for a time complexity of the form $O(f(k) \cdot n^c)$, where k is the parameter, n is the input size and c some constant.

Given an (undirected or directed) graph G, we let $V(G)$ denote its vertex set and $E(G)$ its edge set. If $V' \subseteq V(G)$ then $G[V']$ denotes the subgraph of G induced by the vertex set V'. We usually reserve the symbols n and m to denote, respectively, the number of vertices and the number of edges in a graph. We let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{N}_0$ the set $\mathbb{N} \cup \{0\}$. The symbol $\mathbb{Q}$ denotes the set of rationals and $\mathbb{Q}^{\geq a}$, the set of rationals at least a. For a positive integer n, we let $[n]$ denote the set $\{1, \ldots, n\}$. Logarithms to base 2 are denoted by log and to base e by ln.

1.5 Aims and Organization of the Thesis

As discussed in Section 1.2, an NP-optimization problem usually admits several parameterizations. One of the main aims of this thesis is to study different parameterizations of NP-optimization problems with the intent of identifying parameterizations that are most useful from a practical point of view. In Chapters 2 and 3, we try to develop a general theory for parameterizing or approximating a problem beyond a default or best-known bound.

We follow this up (in Chapters 4 through 7) with a study of the parameterized complexity of several concrete NP-optimization problems. We often consider different ways of parameterizing a problem and investigate how this changes the (parameterized) complexity of the problem. A detailed organization of the thesis follows.

In Chapter 2 we study NP-optimization problems which have nontrivial lower bounds on the size of their optimal solutions. A lower bound is nontrivial if it is an unbounded function of the input size. We show that a number of NP-optimization problems and, in particular problems in MAX SNP, have this property and that the standard parameterized version of these problems is trivially in FPT. We argue that the natural parameter for such problems is the quantity above the guaranteed lower bound. But parameterizing above any lower bound may not be interesting, for if a lower bound is "loose" then even an "above-guarantee" parameterization may trivially be in FPT. This motivates the introduction of tight lower bounds. Problems parameterized beyond tight bounds may be in FPT or W-hard, and we give examples of both. However we show that parameterizations "sufficiently" beyond tight bounds are not in FPT, unless P = NP, for a large class of NP-optimization problems.

In Chapter 3, we consider related questions in the approximation algorithms setting. If an NP-optimization problem admits an α-approximation algorithm, what then is the complexity of obtaining an $(\alpha+\epsilon)$-approximate solution? Since α could be the approximation lower-bound for the problem, the algorithm in question could possibly have a worst-case exponential-time complexity. But the challenge is to obtain moderately exponential-time algorithms, whose running time is possibly a function of ϵ and the input-size, that deliver $(\alpha+\epsilon)$-approximate solutions. We discuss a technique that allows us to obtain such algorithms for a class of NP-optimization problems.

In Chapter 4, we study the approximation and parameterized complexity of a set of problems where one is required to obtain the largest Kőnig subgraph of a given graph. A graph G is called Kőnig if the size of a maximum matching is equal to that of a minimum vertex cover of G. One of the problems we study is Kőnig Vertex Deletion: given a graph $G = (V, E)$ and a nonnegative integer k, does there exist a set of at most k vertices whose deletion yields a Kőnig graph? A vertex set which has the property that deleting it results in a Kőnig graph is called a Kőnig vertex deletion set. We will see that this problem is closely related to the Above Guarantee Vertex Cover problem (Section 1.2) and that there are interesting structural relations between vertex covers, matchings and Kőnig vertex deletion sets.

In Chapter 5 we consider a problem called Unique Coverage which has applications in wireless networking and radio broadcasting and is also a natural generalization of the well-known Max Cut problem. Like the Longest Common Subsequence problem (discussed in Section 1.2), this

problem admits several parameterizations and we show that all of them, except the standard version, are unlikely to be fixed-parameter tractable. Using techniques from Extremal Combinatorics we show that this problem has a 4^k kernel. We then use the color-coding technique to show that a weighted variant of this problem, called Budgeted Unique Coverage, is in FPT. For derandomizing our algorithms, we use k-perfect hash families. However, we can also use an interesting variation of k-perfect hash families where, for every s-element subset S of the universe and every k-element subset X of S, there exists a function that maps X injectively and maps the remaining elements of S into a different range. We show the existence of such families of size smaller than the best-known s-perfect hash families using the probabilistic method. However an explicit construction of such families of the size promised by the probabilistic method is open.

In Chapters 6 and 7, we study the parameterized complexity of some NP-optimization problems on directed graphs. In Chapter 6 we consider the parameterized complexity of the following problem: Given a hereditary property $\mathcal{P}$ on digraphs, an input digraph D and a positive integer k, does D have an induced subdigraph on k vertices with property $\mathcal{P}$? We completely characterize hereditary properties for which this induced subgraph problem is W[1]-complete for two classes of directed graphs: general directed graphs and oriented graphs. We also characterize those properties for which the induced subgraph problem is W[1]-complete for general directed graphs but fixed parameter tractable for oriented graphs.

In Chapter 7 we study a directed analog of the Full Degree Spanning Tree problem where, given a digraph D and a nonnegative integer k, the goal is to construct a spanning out-tree of D with at least k vertices of full out-degree. We show that this problem is W[1]-hard on two basic digraph classes: directed acyclic digraphs and strongly connected digraphs. In the dual version, called Reduced Degree Spanning Tree, we parameterize from the other extreme. Here the goal is to construct a spanning out-tree with at most k vertices of reduced out-degree. We show that this problem is fixed-parameter tractable and admits a problem kernel with at most $8k$ vertices on strongly connected digraphs and $O(k^2)$ vertices on general digraphs. We also give an algorithm for this problem on general digraphs with running time $O^*(5.942^k)$.

Finally in Chapter 8 we summarize the results of this thesis and mention some open problems.

Chapter 2

Parameterizing From Default Values

In this chapter we consider parameterizations of NP-optimization problems with the property that their optimal solution size is either bounded below or bounded above by an unbounded function of the input size. We argue that for such optimization problems, the natural parameter is the quantity above the default lower bound or below the default upper bound. That is, one must either consider the *above-guarantee* or *below-guarantee* parameterizations relative to the lower or upper bound, respectively.

Apart from showing that there are many natural NP-optimization problems which exhibit this property, we introduce the notion of "tight" upper and lower bounds and show that parameterizations above or below tight bounds may be either in FPT or W-hard. Moreover, if we parameterize "sufficiently" above or below tight bounds, then these parameterized versions are not fixed-parameter tractable unless P = NP, for a subclass of NP-optimization problems.

2.1 Why Parameterize From Default Values?

Recall from Chapter 1 that the standard parameterized version Q_{par} of an NP-optimization problem Q is defined as follows.

Input: A tuple (I, k), where I is an instance of Q and k is a non-negative integer, the parameter.

Question:

- Is $\text{opt}(I) \geq k$? (Q is a maximization problem).
- Is $\text{opt}(I) \leq k$? (Q is a minimization problem).

Although the standard parameterized version is the most popularly studied parameterization of NP-optimization problems, there are many problems for which the standard version is trivially fixed-parameter tractable. Consider for instance the problem Max c-Sat. An instance of this problem consists of a Boolean Cnf formula with at most c literals per clause and the objective is

to find an assignment which satisfies the maximum number of clauses. This problem is known to be NP-complete for $c \geq 2$. It is well-known that if ϕ is a Boolean CNF formula with m clauses then there exists an assignment that satisfies at least $\lceil m/2 \rceil$ clauses and that such an assignment can be found in time $O(|\phi|)$ (see [104]).

Now consider what this means for the standard parameterized version of MAX c-SAT, for $c \geq 2$. An instance of the parameterized version consists of a tuple (ϕ, k), where ϕ is a Boolean CNF formula with m clauses and at most c literals per clause and k is a nonnegative integer. The question is whether there exists an assignment that satisfies at least k clauses of ϕ. If $k \leq \lceil m/2 \rceil$, then there exists an assignment which satisfies at least k clauses (because there is one that satisfies $\lceil m/2 \rceil$ clauses) and such an assignment can be obtained in linear time. We therefore answer YES and output the assignment. Otherwise, $k > \lceil m/2 \rceil$ and since every clause has at most c literals, we have $|\phi| \leq 2kc$. We now use a brute-force algorithm that looks at all possible assignments to the variables of ϕ (which are at most $2kc$ in number) and check whether any of these assignments satisfies at least k clauses of ϕ. This brute-force algorithm runs in time $O(2^{2kc} \cdot |\phi|)$ and is, by definition, an FPT-algorithm for MAX c-SAT. Note, however, that when the brute-force algorithm is applied, k, and hence the running time is large for all practical purposes.

A similar state-of-affairs exists for several other problems: MAX CUT, PLANAR INDEPENDENT SET and MAX ACYCLIC SUBGRAPH to name a few. All these problems have some nontrivial lower bound for the optimum value which is exploited to give a trivial fixed-parameter algorithm for the standard parameterized version of the problem. When the parameter value is below the default lower bound, the answer is NO; otherwise, the standard brute-force algorithm itself gives a fixed-parameter algorithm since in this case, the parameter value is considerably large compared to the input size. However, as we observed before, these algorithms are not necessarily practical.

To deal with this problem, Mahajan and Raman [98] considered a different parameterization of MAX SAT and MAX CUT, where the parameter is the *difference* between the optimum value and the guaranteed lower bound. They showed that both these problems are FPT under this parameterization as well. More recently, the "above-guarantee" versions of LINEAR ARRANGEMENT [76] and MINIMUM PROFILE [82, 77] have been shown to be in FPT. Our first observation in this paper is that not just these problems, but all optimization problems in MAX SNP have a nontrivial lower bound for the optimum value. We prove that the above-guarantee parameterization with respect to this lower bound is fixed-parameter tractable. We also observe that approximation algorithms give nontrivial lower or upper bounds on the solution size. We show that the above or below-guarantee question with respect to these bounds is fixed-parameter tractable whenever the stan-

dard parameterized versions of these problems is fixed-parameter tractable. This is dealt with in Section 2.3 after some definitions in Section 2.2 related to optimization problems.

We next show that parameterizing above any nontrivial lower bound may not be interesting because, for some nontrivial lower bounds, the above-guarantee question may still be trivially FPT. This motivates us to define what are known as tight lower bounds (Section 2.4).

Tight upper bounds can be analogously defined. Note that most of the problems we discussed also have nontrivial upper bounds for the optimum value, and another natural parameterization is to parameterize below the upper bound. For example, Max Sat has the number of clauses in the input formula, m, as an upper bound; the upper bound for Max Cut is the number of edges m in the graph. The natural below-guarantee parameterized questions are: can you satisfy all but k clauses and is there a cut of size at least $m-k$? The first is FPT for 2-Cnf Sat [116], and hard (not in FPT unless P = NP) for c-Cnf Sat for $c \geq 3$ [98]. The second problem is the well-known Odd Cycle Transversal problem which was shown to be FPT in [117]. Several vertex (edge)-deletion problems, for example, König Vertex/Edge Deletion [102], Feedback Vertex Set [115], Planar Vertex Deletion [100], fit in the below-guarantee framework.

In Section 2.5 we show some problems for which the above-guarantee or below-guarantee version is unlikely to be in FPT. In Section 2.6 we show that if we parameterize "sufficiently above" tight lower bounds then the above-guarantee question becomes hard (unless P = NP) for a number of NP-optimization problems. We show a similar result for the below-guarantee question with respect to tight upper bounds. Finally in Section 2.8 we list a number of interesting research directions.

2.2 Preliminaries

An NP-optimization (NPO) problem Q is a four-tuple $Q = \{\mathscr{I}, S, V, \text{opt}\}$, where

1. $\mathscr{I}$ is the set of input instances. (Without loss of generality, $\mathscr{I}$ can be recognized in polynomial time.)

2. $S(x)$ is the set of feasible solutions for the input $x \in \mathscr{I}$.

3. V is a polynomial-time computable function called the *cost function* and for each $x \in \mathscr{I}$ and $y \in S(x)$, $V(x, y) \in \mathbb{N}$.

4. opt $\in \{\max, \min\}$.

5. The following decision problem $\tilde{Q}$ (called the *underlying decision problem of* Q) is in NP: Given $x \in \mathscr{I}$ and an integer k, does there exist

a feasible solution $y \in S(x)$ such that $V(x, y) \geq k$, when Q is a maximization problem (or, $V(x, y) \leq k$, when Q is a minimization problem).

From the above definition, it follows that the optimum value of every NP-optimization problem is bounded below by 1. We call a lower bound *trivial* if it is at most a constant. A *nontrivial lower bound* for an NP-optimization problem Q is an unbounded function $f : \mathbb{N} \to \mathbb{N}$ such that for all instances x of Q, we have $\text{opt}(x) \geq f(|x|)$. A *nontrivial upper bound* for Q is an unbounded function $g : \mathbb{N} \to \mathbb{N}$ such that for all instances x of Q, we have $\text{opt}(x) \leq g(|x|)$.

The class MAX SNP was defined by Papadimitriou and Yannakakis [109] using logical expressiveness. They showed that a number of interesting optimization problems such as Max 3-Sat, Independent Set-B, Max Cut, Max k-Colorable Subgraph lie in this class. They also introduced the notion of completeness for MAX SNP by a reduction known as the *L-reduction* which we define next.

Let Q_1 and Q_2 be two optimization (maximization or minimization) problems. We say that Q_1 L-reduces to Q_2 if there exist polynomial-time computable functions f, g, and constants $\alpha, \beta > 0$ such that for each instance I_1 of Q_1:

1. $f(I_1) = I_2$ is an instance of Q_2, such that $\text{opt}(I_2) \leq \alpha \cdot \text{opt}(I_1)$.

2. Given any solution y_2 of I_2, g maps (I_2, y_2) to a solution y_1 of I_1 such that $|V(I_1, y_1) - \text{opt}(I_1)| \leq \beta \cdot |V(I_2, y_2) - \text{opt}(I_2)|$

We call such an L-reduction from Q_1 to Q_2 an (f, g, α, β)-reduction.

An NP-optimization problem Q is MAX SNP-hard if every problem in the class MAX SNP L-reduces to Q. A problem Q is MAX SNP-complete, if Q is in MAX SNP and is MAX SNP-hard. Some example MAX SNP-complete problems are

1. Max c-Sat, for any constant c,

2. Independent Set-B,

3. Vertex Cover-B,

4. Dominating Set-B,

5. Max Cut,

6. Max Directed Cut,

7. Max k-Colorable Subgraph.

Cai and Chen [26] established that the standard parameterized version of all maximization problems in the class MAX SNP are fixed-parameter tractable. In the next section, we show that for all problems in MAX SNP, a certain above-guarantee question is also fixed-parameter tractable.

2.3 Parameterizing From Default Values

The main objective of this section is to show that there exist broad classes of NP-optimization problems in which every problem has a nontrivial lower or upper bound on the optimum solution size and that the above or below guarantee question with respect to these bounds is in FPT. Recall that a lower or upper bound is nontrivial if it is an unbounded function of the input size.

We first show that every problem in the class MAX SNP has a nontrivial lower bound on the optimal solution size and that the parameterized above- or below-guarantee question with respect to this lower bound is in FPT.

2.3.1 Parameterizing Above the 3-Sat Lower Bound

Consider the problem Max 3-Sat which is complete for the class MAX SNP. An instance of Max 3-Sat is a Boolean formula f in conjunctive normal form with at most three literals per clause. As already stated, any Boolean formula with m clauses has at least $\lceil m/2 \rceil$ satisfiable clauses. Using this, we show the following generalization.

Proposition 2.1. *If Q is in* MAX SNP*, then for each instance I of Q there exists a positive number γ_x such that $\gamma_x \leq opt(x)$. Further, if Q is* NP*-hard, then the function $\gamma : x \to \gamma_x$ is unbounded, assuming* P $\neq$ NP*.*

Proof. Let Q be a problem in MAX SNP and let (f, g, α, β) be an L-reduction from Q to Max 3-Sat. Then for an instance I of Q, $f(x)$ is an instance of Max 3-Sat such that $\text{opt}(f(x)) \leq \alpha \cdot \text{opt}(x)$. If $f(x)$ is a formula with m clauses, then $\lceil m/2 \rceil \leq \text{opt}(f(x))$ and therefore $\text{opt}(x)$ is bounded below by $\lceil m/2 \rceil / \alpha$. This proves that each instance x of Q has a lower bound. We can express this lower bound in terms of the parameters of the L-reduction. Since $f(x)$ is an instance of Max 3-Sat, we can take the size of $f(x)$ to be m. Then $\gamma_x = |f(x)|/2\alpha = m/2\alpha$. Further, note that if m is not unbounded, then we can solve Q in polynomial time via this reduction. □

Note that this lower bound γ_x depends on the complete problem to which we reduce Q. By changing the complete problem, we might construct different lower bounds for the problem at hand. It is also conceivable that there exist more than one L-reduction between two optimization problems. Different L-reductions could give different lower bounds. Thus the

polynomial-time computable lower bound that we exhibit in Proposition 2.1 is a special lower bound obtained from a specific L-reduction to a specific complete problem (MAX 3-SAT) for the class MAX SNP. Call the lower bound of Proposition 2.1 a MAX 3-SAT-lower bound for the problem Q.

Consider the above-guarantee parameterized version L of MAX 3-SAT: the language L consists of tuples (f, k) such that f is an instance of MAX 3-SAT and there exists an assignment satisfying at least $k + \lceil m/2 \rceil$ clauses of the formula f. This problem is fixed-parameter tractable with respect to parameter k [98] and using this we prove the following.

Theorem 2.1. *Let Q be a maximization problem in* MAX SNP *and let (f, g, α, β) an L-reduction from Q to* MAX 3-SAT. *For an instance x of Q, let γ_x represent the* MAX 3-SAT*-lower bound of x. Then the following problem is in* FPT*:*

$$L_Q = \{(x, k) : \ x \text{ is an instance of } Q \text{ and } opt(x) \geq \gamma_x + k\}$$

Proof. We make use of the fact that there exists a fixed-parameter tractable algorithm $\mathscr{A}$ for MAX 3-SAT which takes as input, a pair of the form (ψ, k), and in time $O(|\psi| + h(k))$, returns YES if there exists an assignment to the variables of ψ that satisfies at least $\lceil m/2 \rceil + k$ clauses, and NO otherwise. See [98, 108] for such algorithms.

Consider an instance (x, k) of L_Q. Then $f(x)$ is an instance of MAX 3-SAT. Let $f(x)$ have m clauses. Then the guaranteed lower bound for the instance x of Q, $\gamma_x = m/2\alpha$, and $\text{opt}(f(x)) \leq \alpha \cdot \text{opt}(x)$. Apply algorithm $\mathscr{A}$ on input $(f(x), k\alpha)$. If $\mathscr{A}$ outputs YES, then $\text{opt}(f(x)) \geq m/2 + k\alpha$, implying $\text{opt}(x) \geq m/2\alpha + k = \gamma_x + k$. Thus $(x, k) \in L_Q$.

If $\mathscr{A}$ answers NO, then $\lceil m/2 \rceil \leq \text{opt}(f(x)) < \lceil m/2 \rceil + k\alpha$. Apply algorithm $\mathscr{A}$ on the inputs $(f(x), 1), (f(x), 2), \ldots, (f(x), k\alpha)$, one by one, to obtain $\text{opt}(f(x))$. Let $c' = \text{opt}(f(x))$. Then use algorithm g of the L-reduction to obtain a solution to x with cost c. By the definition of L-reduction, we have $|c - \text{opt}(x)| \leq \beta \cdot |c' - \text{opt}(f(x))|$. But since $c' = \text{opt}(f(x))$, it must be that $c = \text{opt}(x)$. Therefore we simply need to compare c with $\gamma_x + k$ to check whether $(x, k) \in L_Q$.

The total time complexity of the above algorithm is $O(k\alpha \cdot (|f(x)| + h(k\alpha)) + p_1(|x|) + p_2(|f(x)|))$, where $p_1(\cdot)$ is the time taken by algorithm f to transform an instance of Q to an instance of MAX 3-SAT, and $p_2(\cdot)$ is the time taken by g to output its answer. Thus the algorithm that we outlined is indeed an FPT-algorithm for L_Q. □

Note that the proof of Proposition 2.1 also shows that every minimization problem in MAX SNP has a MAX 3-SAT-lower bound. For minimization problems whose optimum is bounded below by some function of the input, it makes sense to ask how far removed the optimum is with respect to the lower bound. The parameterized question asks whether for a given input x, $\text{opt}(x) \leq \gamma_x + k$, with k as parameter.

Theorem 2.2. *Let Q be a minimization problem in* MAX SNP *and let (f, g, α, β) an L-reduction from Q to* MAX 3-SAT. *For an instance x of Q, let γ_x represent the* MAX 3-SAT*-lower bound of x. Then the following problem is in* FPT*:*

$$L_Q = \{(x, k) : \ x \text{ is an instance of } Q \text{ and } opt(x) \leq \gamma_x + k\}.$$

Proof. As before, let $\mathscr{A}$ be an FPT-algorithm for MAX 3-SAT which takes as input, a pair of the form (ψ, k), and in time $O(|\psi| + h(k))$, returns YES if there exists an assignment to the variables of ψ that satisfies at least $\lceil m/2 \rceil + k$ clauses, and NO otherwise. Let (x, k) be an instance of L_Q and $f(x)$ the instance of MAX 3-SAT with m clauses. Then $\gamma_x = m/2\alpha$ and $\text{opt}(f(x)) \leq \alpha \cdot \text{opt}(x)$. Apply algorithm $\mathscr{A}$ on input $(f(x), (k+1) \cdot \alpha)$. If $\mathscr{A}$ outputs YES, then $\text{opt}(f(x)) \geq m/2 + (k+1) \cdot \alpha$, implying $\text{opt}(x) \geq m/2\alpha + k + 1 = \gamma_x + k + 1$. In this case, $(x, k) \notin L_Q$ and we return NO. If $\mathscr{A}$ answers NO, then $\lceil m/2 \rceil \leq \text{opt}(f(x)) < \lceil m/2 \rceil + (k+1) \cdot \alpha$. Apply algorithm $\mathscr{A}$ $(k+1) \cdot \alpha - 1$ times on inputs $(f(x), 1), (f(x), 2), \ldots, (f(x), (k+1) \cdot \alpha - 1)$ to obtain $\text{opt}(f(x))$. Obtain $\text{opt}(x)$ as described in the proof of Theorem 2.1 and decide the instance (x, k) appropriately. □

Examples of minimization problems in MAX SNP include VERTEX COVER-B and DOMINATING SET-B which are, respectively, the restriction of the VERTEX COVER and the DOMINATING SET problems to graphs whose vertex degree is bounded by B.

2.3.2 Guarantees Defined by Approximation Algorithms

If an NP-maximization problem admits an α-approximation algorithm, $0 < \alpha < 1$, then $\alpha \cdot \text{OPT}$ is a nontrivial polynomial-time computable lower-bound on the solution size, where OPT denotes the optimal solution size. For minimization problems that admit an α-approximation algorithm we have $\alpha > 1$ and $\alpha \cdot \text{OPT}$ is a polynomial-time computable upper-bound on the solution size. One can parameterize above or below these nontrivial guaranteed values. We show that if (1) the NP-optimization problem is polynomially bounded, (2) its standard parameterized version is in FPT, and (3) α is a constant, then the $\alpha \cdot \text{OPT} + k$ question (for maximization problems) and the $\alpha \cdot \text{OPT} - k$ question (for minimization problems) are in FPT.

Theorem 2.3. *Let Q be an* NPO *problem which admits a constant-factor α-approximation algorithm such that its standard parameterized version is in* FPT*. Then the following problems are in* FPT*:*

$$L_1 = \{(I, k) : \max(I) \geq \alpha \cdot \text{OPT} + k\} \quad \text{if } Q \text{ is a maximization problem;}$$
$$L_2 = \{(I, k) : \min(I) \leq \alpha \cdot \text{OPT} - k\} \quad \text{if } Q \text{ is a minimization problem.}$$

Proof. Suppose Q is a maximization problem and let (I, k) be an instance of L_1. Then $\max(I) \geq \alpha \cdot \text{OPT} + k$ if and only if $k \leq (1-\alpha) \cdot \text{OPT}$, that is, if and only if $k/(1-\alpha) \leq \text{OPT}$. Since the standard parameterized version of Q is in FPT, deciding whether $k/(1-\alpha) \leq \text{OPT}$ is in FPT. The proof when Q is a minimization problem is similar. □

2.4 Tight Lower and Upper Bounds

For an optimization problem, the question of whether the optimum is at least "lower bound + k", for some lower bound and k as parameter, is not always interesting because if the lower bound is "loose" then the problem is trivially fixed-parameter tractable. For instance, the following "above-guarantee" version of Max Cut is trivially in FPT. Given a connected graph G on m edges, does G have a cut of size at least $m/2 + k$? By Poljak and Turzík [111], any connected graph G with m edges and n vertices has a cut of size at least $m/2 + \lceil (n-1)/4 \rceil$, and so if $k \leq \lceil (n-1)/4 \rceil$, we answer yes. Otherwise $n \leq 4k$ and a brute-force algorithm that considers all possible vertex partitions is an FPT-algorithm.

We therefore examine the notion of a *tight lower bound* and the corresponding above-guarantee question. A tight lower bound is essentially the best possible lower bound on the optimum solution size. For the Max Sat problem, this lower bound is $m/2$: if ϕ is an instance of Max Sat, then $\text{opt}(\phi) \geq m/2$, and there are infinitely many instances for which the optimum is exactly $m/2$. This characteristic motivates the next definition.

Definition 2.1 (Tight Lower Bound)**.** A function $f : \mathbb{N} \to \mathbb{N}$ is a *tight lower bound* for an NP-optimization problem $Q = \{\mathscr{I}, S, V, \text{opt}\}$ if the following conditions hold:

1. $f(|I|) \leq \text{opt}(I)$ for all $I \in \mathscr{I}$.
2. There exists an infinite family of instances $\mathscr{I}' \subseteq \mathscr{I}$ such that $\text{opt}(I) = f(|I|)$ for all $I \in \mathscr{I}'$.

Note that we define the lower bound to be a function of the *input size* rather than the input itself. This is in contrast to the lower bound in Proposition 2.1 which depends on the input instance. We can define the notion of a tight upper bound analogously.

Definition 2.2 (Tight Upper Bound)**.** A function $g : \mathbb{N} \to \mathbb{N}$ is a *tight upper bound* for an NP-optimization problem $Q = \{\mathscr{I}, S, V, \text{opt}\}$ if the following conditions hold:

1. $\text{opt}(I) \leq g(|I|)$ for all $I \in \mathscr{I}$.
2. There exists an infinite family of instances $\mathscr{I}' \subseteq \mathscr{I}$ such that $\text{opt}(I) = g(|I|)$ for all $I \in \mathscr{I}'$.

Some example optimization problems which have tight lower and upper bounds are given below. The abbreviations TLB and TUB stand for tight lower bound and tight upper bound, respectively.

MAX EXACT c-SAT

INSTANCE A Boolean formula F with n variables and m clauses with each clause having *exactly* c distinct literals.

QUESTION Find the maximum number of simultaneously satisfiable clauses.

BOUNDS TLB $= (1 - 1/2^c)m$; TUB $= m$.

The expected number of clauses satisfied by the random assignment algorithm is $(1 - 1/2^c)m$; hence the lower bound. To see tightness, note that if $\phi(x_1, \ldots, x_c)$ denotes the EXACT c-SAT formula comprising of all possible combinations of c variables, then ϕ has 2^c clauses of which exactly $2^c - 1$ clauses are satisfiable. By taking disjoint copies of this formula one can construct EXACT c-SAT instances of arbitrary size with exactly $(1-1/2^c)m$ satisfiable clauses.

MAX LIN-2

INSTANCE A system of m linear equations modulo 2 in n variables, together with positive weights w_i, $1 \leq i \leq m$.

QUESTION Find an assignment to the variables that maximizes the total weight of the satisfied equations.

BOUNDS TLB $= W/2$, where $W = \sum_{i=1}^{m} w_i$; TUB $= W$.

If we use $\{+1, -1\}$-notation for Boolean values with -1 corresponding to true then we can write the ith equation of the system as $\prod_{j \in \alpha_i} x_j = b_i$, where each α_i is a subset of $[n]$ and $b_i \in \{+1, -1\}$. To see that we can satisfy at least half the equations in the weighted sense, we assign values to the variables sequentially and simplify the system as we go along. When we are about to give a value to x_j, we consider all equations reduced to the form $x_j = b$, for a constant b. We choose a value for x_j satisfying at least half (in the weighted sense) of these equations. This procedure of assigning values ensures that we satisfy at least half the equations in the weighted sense. A tight lower bound instance, in this case, is a system consisting of pairs $x_j = b_i, x_j = \bar{b}_i$, with each equation of the pair assigned the same weight. See [81] for more details.

MAX INDEPENDENT SET-B

INSTANCE A graph G with n vertices such that the degree of each vertex is bounded by B.

QUESTION Find a maximum independent set of G.

BOUNDS TLB $= n/(B + 1)$; TUB $= n$.

A graph whose vertex degree is bounded by B can be colored using $B+1$ colors, and in any valid coloring of the graph, the vertices that get the same color form an independent set. By the pigeonhole principle, there exists an independent set of size at least $n/(B+1)$. The complete graph K_{B+1} on $B+1$ vertices has an independence number of $n/(B+1)$. By taking disjoint copies of K_{B+1} one can construct instances of arbitrary size with independence number exactly $n/(B+1)$.

MIN DOMINATING SET-B

INSTANCE A graph G with n vertices such that the degree of each vertex is bounded by B.

QUESTION Find a minimum dominating set of G.

BOUNDS TLB $= n/(B+1)$; TUB $= n$.

Observe that any vertex can dominate at most $B+1$ vertices (including itself) and thus any dominating set has size at least $n/(B+1)$. A set of r disjoint copies of K_{B+1} has a minimum dominating set of size exactly $r = n/(B+1)$. The upper bound of n is met by the class of empty graphs.

MIN VERTEX COVER-B

INSTANCE A graph G with n vertices and m edges such that the degree of each vertex is at least one and at most B.

QUESTION Find a minimum vertex cover of G.

BOUNDS TLB $= m/B$; TUB $= nB/(B+1)$.

Each vertex can cover at most B edges and so any vertex cover has size at least m/B. This bound is tight for a set of disjoint copies of $K_{1,B}$'s. The upper bound follows from the fact that an independent set in such a graph has size at least $n/(B+1)$. The upper bound is met by a disjoint collection of K_{B+1}'s.

MAX PLANAR INDEPENDENT SET

INSTANCE A planar graph G with n vertices and m edges.

QUESTION Find a maximum independent set of G.

BOUNDS TLB $= n/4$; TUB $= n$.

A planar graph is 4-colorable, and in any valid 4-coloring of the graph, the vertices that get the same color form an independent set. By the pigeonhole principle, there exists an independent set of size at least $n/4$. A disjoint set of K_4's can be use to construct arbitrary sized instances with independence number exactly $n/4$.

MAX ACYCLIC DIGRAPH

INSTANCE A directed graph G with n vertices and m arcs.

QUESTION Find a maximum arc-induced acyclic subgraph of G.

BOUNDS TLB $= m/2$; TUB $= m$.

To see that any digraph with m arcs has an acyclic subgraph of size $m/2$, place the vertices $v_1, \ldots, v_n$ of G on a line in that order with arcs (v_i, v_j), $i < j$, drawn above the line and arcs (v_i, v_j), $i > j$, drawn below the line. Clearly, by deleting all arcs either above or below the line we obtain an acyclic digraph. By the pigeonhole principle, one of these two sets must have size at least $m/2$. To see that this bound is tight, consider the digraph D on n vertices: $v_1 \leftrightarrows v_2 \leftrightarrows v_3 \leftrightarrows \ldots \leftrightarrows v_n$ which has a maximum acyclic digraph of size exactly $m/2$. Since n is arbitrary, we have an infinite set of instances for which the optimum matches the lower bound exactly.

MAX PLANAR SUBGRAPH

INSTANCE A connected graph G with n vertices and m edges.

QUESTION Find an edge-subset E' of maximum size such that $G[E']$ is planar.

BOUNDS TLB $= n - 1$; TUB $= 3n - 6$.

Any spanning tree of G has $n-1$ edges; hence any maximum planar subgraph of G has at least $n - 1$ edges. This bound is tight as the family of all trees achieves this lower bound. An upper bound is $3n - 6$ which is tight since for each n, a maximal planar graph on n vertices has exactly $3n - 6$ edges.

MAX CUT

INSTANCE A graph G with n vertices, m edges and c components.

QUESTION Find a maximum cut of G.

BOUNDS TLB $= m/2 + \lceil (n - c)/4 \rceil$; TUB $= m$.

The lower bound for the cut size was proved by Poljak and Turzík [111]. This bound is tight for complete graphs. The upper bound is tight for bipartite graphs.

MIN LINEAR ARRANGEMENT

INSTANCE An undirected graph $G = (V, E)$ with n vertices and m edges.

QUESTION Find a one-to-one mapping $\sigma : V \to \{1, 2, \ldots, |V|\}$ such that the function $\sum_{\{u,v\} \in E} |\sigma(u) - \sigma(v)|$ is a minimum.

BOUNDS TLB $= m$; TUB $= n(n-1)(n+1)/6 = \binom{n+1}{3}$.

The minimum value of $\sum_{\{u,v\}\in E} |\sigma(u) - \sigma(v)|$ over all one-to-one maps σ is denoted by $\text{ola}(G)$. Clearly $|\sigma(u) - \sigma(v)| \geq 1$ for any mapping σ and hence $\sum_{\{u,v\}\in E} |\sigma(u) - \sigma(v)| \geq m$. This lower bound is tight for the set of paths. For any n-vertex graph G, $\text{ola}(G) \leq \text{ola}(K_n)$, where K_n denotes the complete graph on n vertices. It can be easily seen that for the case of K_n, the value of the objective function remains the same for all one-to-one maps σ and that $\text{ola}(K_n) = 1 \cdot (n-1) + 2 \cdot (n-2) + \cdots + (n-1) \cdot 1 = n(n-1)(n+1)/6$.

MIN PROFILE

INSTANCE An undirected graph $G = (V, E)$ with n vertices and m edges.

QUESTION Find a one-to-one mapping $\sigma : V \to \{1, 2, \ldots, |V|\}$ such that $\sum_{v\in V} \text{prf}_\sigma(v)$ is a minimum, where

$$\text{prf}_\sigma(v) = \sigma(v) - \min\{\sigma(u) : u \in N[v]\}.$$

BOUNDS TLB $= m$; TUB $= \binom{n}{2}$.

The minimum value of the objective function over all one-to-one maps σ is denoted by $\text{prf}(G)$. This problem is equivalent [17] to the well-known INTERVAL GRAPH COMPLETION problem [64, 82] in that if the minimum number of edges required to be added to a graph to make it interval is k then its profile is $m + k$, where m denotes the number of edges in the graph. Consequently, for any graph $G = (V, E)$,

$$m \leq \text{prf}(G) \leq \binom{n}{2}.$$

Since for each positive integer m there exists an interval graph with m edges, this is a tight lower bound on the profile of a graph. Since complete graphs are interval graphs too, this result also shows that $\binom{n}{2}$ is a tight upper bound. Interestingly, there is a different lower bound if we restrict the input graph to be connected. If G is a connected graph on n vertices then $\text{prf}(G) \geq n - 1$ [94] and this bound is tight since the profile of a path on n vertices is $n - 1$.

Natural questions for maximization problems in the above or below-guarantee framework are whether the languages

$$\begin{aligned} L_{a,\max} &= \{(I, k) : \max(I) \geq \text{TLB}(I) + k\} \\ L_{b,\max} &= \{(I, k) : \max(I) \geq \text{TUB}(I) - k\} \end{aligned}$$

are in FPT. For minimization problems, one can ask whether the following are in FPT.

$$\begin{aligned} L_{a,\min} &= \{(I, k) : \min(I) \leq \text{TLB}(I) + k\} \\ L_{b,\min} &= \{(I, k) : \min(I) \leq \text{TUB}(I) - k\} \end{aligned}$$

The parameterized complexity of such questions are not known for most problems which are known to have tight bounds. To the best of our knowledge, the above-guarantee question has been shown to be in FPT only for the Max Sat and Max c-Sat problems [98] and, more recently, for Linear Arrangement [76] and both versions of Minimum Profile [82, 77].

Since most vertex/edge deletion problems can be cast as below-guarantee parameterized questions, comparatively more results are known about such problems. For instance, for the Independent Set problem a trivial upper bound on the solution size is the number of vertices in the graph. This bound is tight as the family of trivial graphs meets this bound. Given a graph G on n vertices, the question of whether there exists k vertices whose deletion leaves a trivial graph (or equivalently, does G have an independent set on $n-k$ vertices?) is fixed-parameter tractable, being equivalent to the well-known Vertex Cover problem [107]. Other examples include Feedback Vertex Set [115], Directed Feedback Vertex Set [31], Odd Cycle Transversal [117] and Chordal Vertex Deletion [99] which are all of the form: "is there an acyclic (undirected/directed), bipartite and chordal graph, respectively, on $n-k$ vertices?"

An interesting example of a non-graph-theoretic problem parameterized below a guaranteed upper bound is the Min 2-Sat Deletion problem [98].

Min 2-Sat Deletion

Input:	A Boolean 2-CNF formula ϕ with m clauses and a nonnegative integer k.
Parameter:	The integer k.
Question:	Does there exist an assignment that satisfies $m-k$ clauses of ϕ?

This problem has been shown to be fixed-parameter tractable by Razgon et al. [116]. Note that the problem of deciding whether k clauses can be deleted from a c-Cnf formula to make it satisfiable is NP-hard for $c \geq 3$ [98].

In the next section we exhibit problems whose above or below-guarantee parameterized versions are hard.

2.5 Hard Above or Below-Guarantee Problems

We first exhibit two problems whose above-guarantee parameterized versions are hard (not in FPT unless P = NP). To the best of our knowledge, these are the only ones in this category. Consider the problem Min Weight t-connected Spanning Subgraph defined as follows [34]:

Input:	A connected graph G with n vertices and nonnegative integers t and k.
Parameter:	The integer k.

Question: Does there exists a t-vertex-connected spanning subgraph of G with at most k edges?

This problem is NP-complete for undirected graphs for $t \geq 2$ [34] and is easily fixed-parameter tractable as shown below.

Lemma 2.1. *Let $G = (V, E)$ be a simple, connected graph on n vertices and let k, t be nonnegative integers. Then deciding whether G has a t-vertex-connected spanning subgraph with at most k edges can be done in time polynomial in n, for every fixed k.*

Proof. For $t \geq 2$, a t-vertex-connected spanning subgraph of G must have at least n edges. Therefore if $k < n$, answer NO; else, $n \leq k$ and any brute-force algorithm that solves the problem is fixed-parameter tractable. □

As discussed in the proof of Lemma 2.1 above, n is a trivial lower bound for the problem and is tight for the case $t = 2$. The above-guarantee version, however, is not fixed-parameter tractable, unless P = NP.

Theorem 2.4. *Let $G = (V, E)$ be a simple, connected graph on n vertices and let k, t be nonnegative integers. Deciding whether G has a t-vertex connected spanning subgraph with at most $n + k$ edges is not fixed-parameter tractable with respect to parameter k, unless* P = NP.

Proof. Note that for $t = 2$ and $k = 0$ this problem is equivalent to asking whether G has a Hamiltonian cycle and hence if the above-guarantee question can be answered in time $O(f(k) \cdot n^c)$, the Hamiltonian Cycle problem can be solved in polynomial time implying P = NP. □

Next consider the Bounded Degree Min Spanning Tree [67] problem.

Input: A connected graph $G = (V, E)$ with edge costs $w : E \to \mathbb{Z}^+$ and nonnegative integers c and k.

Parameter: The integer k.

Question: Does there exist a spanning subgraph T of total edge-weight at most k such that each vertex in T has degree at most c?

Since the total weight of any spanning tree is at least $n - 1$, we have the following result.

Lemma 2.2. *The* Bounded Degree Min Spanning Tree *problem is fixed-parameter tractable with respect to parameter k.*

The above-guarantee version (Does G have a spanning tree with total weight at most $n - 1 + k$ and vertex degrees bounded by c?) is again not fixed-parameter tractable unless P = NP as the case $c = 2$ and $k = 0$ reduces to solving the Hamiltonian Path problem.

Theorem 2.5. *Given a connected graph $G = (V, E)$ with edge costs $w : E \to \mathbb{Z}^+$ and positive integers c and k, deciding whether G has a spanning tree with weight at most $n - 1 + k$ and vertex degrees bounded by c is not fixed-parameter tractable with respect to k unless* P = NP.

We next consider a problem parameterized below a tight upper bound, called Wheel-free Deletion, that is W[2]-hard [95]. A vertex v in a graph G is said to be *universal* if v is adjacent to all vertices of G. A *wheel* is a graph W that has a universal vertex v such that $W - v$ is a cycle. A graph is wheel-free if no subgraph of G is a wheel. The class of wheel-free graphs is hereditary [86] and poly-time recognizable [95]. The Wheel-free Vertex/Edge Deletion problem is defined below:

Input:	A graph $G = (V, E)$ with n vertices and m edges and a nonnegative integer k.
Parameter:	The integer k.
Question:	Does G have an induced subgraph on $n - k$ vertices/$m - k$ edges that is wheel-free?

Note that for all n and m there exist wheel-free graphs on n vertices or m edges. The tight upper bound is witnessed by the class of paths, for instance.

In [95], the Wheel-free Vertex/Edge Deletion problems were shown to be W[2]-hard by a reduction from Hitting Set. This is one of the few graph-modification problems known to be hard.

2.6 Parameterizing Sufficiently Beyond Default Values

In this section, we study somewhat different, but related, parameterized questions: Given an NP-maximization problem Q with a tight lower and a tight upper bound, denoted by TLB and TUB, respectively, what is the parameterized complexity of the following questions?

$$Q_{a,\max}(\epsilon) = \{(I, k) : \max(I) \geq \text{TLB}(I) + \epsilon \cdot |I| + k\} \quad (2.1)$$

$$Q_{b,\max}(\epsilon) = \{(I, k) : \max(I) \geq \text{TUB}(I) - \epsilon \cdot |I| - k\} \quad (2.2)$$

Here $|I|$ denotes the input size, ϵ is some fixed positive rational, k is the parameter and a and b denote, respectively, the above and below-guarantee version of the problem. For NP-minimization problems, the corresponding questions are:

$$Q_{a,\min}(\epsilon) = \{(I, k) : \min(I) \leq \text{TLB}(I) + \epsilon \cdot |I| + k\} \quad (2.3)$$

$$Q_{b,\min}(\epsilon) = \{(I, k) : \min(I) \leq \text{TUB}(I) - \epsilon \cdot |I| - k\} \quad (2.4)$$

In Theorem 2.6, we show that Problems 2.1 and 2.3 are not fixed-parameter tractable for a certain class of problems, unless P = NP. Theorem 2.7 establishes this result for Problems 2.2 and 2.4.

To define the class of optimization problems for which we establish the hardness result in Theorem 2.6, we need some definitions. To motivate these, we start with an overview of the proof for maximization problems (Problem 2.1 above). Assume that for some ϵ in the specified range, $Q_{a,\max}(\epsilon)$ is indeed in FPT. Now consider an instance (I, s) of the underlying decision version of Q. Here is a P-time procedure for deciding it. If $s \leq \text{TLB}$, then the answer is trivially YES. If s lies between TLB and $\text{TLB} + \epsilon|I|$, then "add" a gadget of suitable size corresponding to the TUB, to obtain an equivalent instance (I', s'). This increases the input size, but since we are adding a gadget whose optimum value matches the upper bound, the increase in the optimum value of I' is more than proportional, so that now s' exceeds $\text{TLB} + \epsilon|I'|$ and we handle this case next. If s already exceeds $\text{TLB} + \epsilon|I|$, then "add" a gadget of suitable size corresponding to the TLB, to obtain an equivalent instance (I', s'). This increases the input size faster than it boosts the optimum value of I', so that now s' exceeds $\text{TLB} + \epsilon|I'|$ by only a constant, say c_1. Use the hypothesized FPT algorithm for $Q_{a,\max}(\epsilon)$ with input (I', c_1) to correctly decide the original question.

To make this proof idea work, we require that the following conditions be met:

1. The NPO problem should be such that "addition" of problem instances is well-defined and that the optimum of the sum is equal to the sum of the optima (see Definition 2.3).

2. There exist gadgets whose addition to a problem instance increases the instance size faster than it does the optimum value (see Property **P1** below).

3. There exist gadgets whose addition to a problem instance increases the optimum value faster than it does the instance size (see Property **P2** below).

4. The gadgets mentioned in points 2 and 3 must be easily constructible (see Definition 2.4).

Definition 2.3 (Partially Additive Problems)**.** An NPO problem $Q = \{\mathscr{I}, S, V, \text{opt}\}$ is said to be *partially additive* if there exists an operator $+$ which maps a pair of instances I_1 and I_2 to an instance $I_1 + I_2$ such that

1. $|I_1 + I_2| = |I_1| + |I_2|$, and

2. $\text{opt}(I_1 + I_2) = \text{opt}(I_1) + \text{opt}(I_2)$.

A partially additive NPO problem that also satisfies the following condition is said to be additive in the framework of Khanna, Motwani et al. [84]: there exists a polynomial-time computable function f that maps any solution s of $I_1 + I_2$ to a pair of solutions s_1 and s_2 of I_1 and I_2, respectively, such that $V(I_1 + I_2, s) = V(I_1, s_1) + V(I_1, s_2)$.

For many graph-theoretic optimization problems, the operator $+$ can be interpreted as disjoint union. Then the problems Max Cut, Max Independent Set-B, Minimum Vertex Cover, Minimum Dominating Set, Maximum Directed Acyclic Subgraph, Maximum Directed Cut are partially additive. For other graph-theoretic problems, one may choose to interpret $+$ as follows: given graphs G and H, $G + H$ refers to a graph obtained by placing an edge between some (possibly arbitrarily chosen) vertex of G and some (possibly arbitrarily chosen) vertex of H. The Max Planar Subgraph problem is partially additive with respect to both these interpretations of $+$. For Boolean formulae ϕ and ψ in conjunctive normal form with disjoint sets of variables, define $+$ as the conjunction $\phi \wedge \psi$. Then the Max Sat problem is easily seen to be partially additive.

Definition 2.4 (Dense Set)**.** Let $Q = \{\mathscr{I}, S, V, \text{opt}\}$ be an NPO problem. A set of instances $\mathscr{I}' \subseteq \mathscr{I}$ is said to be *dense with respect to a set of conditions* $\mathcal{C}$ if there exists a constant $c \in \mathbb{N}$ such that for all closed intervals $[a, b] \subseteq \mathbb{R}^+$ of length $|b - a| \geq c$, there exists an instance $I \in \mathscr{I}'$ with $|I| \in [a, b]$ such that I satisfies all the conditions in $\mathcal{C}$. Further, if such an I can be found in polynomial time (polynomial in b), then $\mathscr{I}'$ is said to be *dense poly-time uniform with respect to* $\mathcal{C}$.

For example, for the Maximum Acyclic Digraph problem, the set of all oriented digraphs (digraphs without 2-cycles) is dense (poly-time uniform) with respect to the condition: $\text{opt}(G) = |E(G)|$.

Let $Q = \{\mathscr{I}, S, V, \text{opt}\}$ be an NP-optimization problem with a tight lower bound $f : \mathbb{N} \to \mathbb{N}$ and a tight upper bound $g : \mathbb{N} \to \mathbb{N}$. We assume that both f and g are increasing and satisfy the following conditions

P1 For all $a, b \in \mathbb{N}$, $f(a + b) \leq f(a) + f(b) + c^*$, where c^* is a constant (positive or negative),

P2 There exists $n_0 \in \mathbb{N}$ and $r \in \mathbb{Q}^+$ such that $g(n) - f(n) > rn$ for all $n \geq n_0$.

Property $P1$ is satisfied by linear functions ($f(n) = an + b$) and by some sub-linear functions such as $\sqrt{n}, \log n, 1/n$. Note that a super-linear function cannot satisfy $P1$.

Now that we have formally defined all the required properties, we can state the theorem precisely.

Theorem 2.6. *Let $Q = \{\mathscr{I}, S, V, opt\}$ be a polynomially bounded* NPO *problem such that the following conditions hold.*

1. *Q is partially additive.*

2. *Q has a tight lower bound (TLB) f, which is increasing and satisfies condition $P1$. The infinite family of instances $\mathscr{I}'$ witnessing the tight lower bound is dense poly-time uniform with respect to the condition $opt(I) = f(|I|)$.*

3. *Q has a tight upper bound (TUB) g, which with f satisfies condition $P2$. The infinite family of instances $\mathscr{I}'$ witnessing the tight upper bound is dense poly-time uniform with respect to the condition $opt(I) = g(|I|)$.*

4. *The underlying decision problem $\tilde{Q}$ of Q is* NP*-hard.*

Let $p := \sup\{r \in \mathbb{Q}^+ : g(n) - f(n) > rn \text{ for all } n \geq n_0\}$ and for $0 < \epsilon < p$, define $Q_a(\epsilon)$ to be the following parameterized problem

$$Q_a(\epsilon) = \{(I,k) : \ opt(I) \geq f(|I|) + \epsilon|I| + k\} \quad \text{for max problems;}$$
$$Q_a(\epsilon) = \{(I,k) : \ opt(I) \leq f(|I|) + \epsilon|I| + k\} \quad \text{for min problems.}$$

If $Q_a(\epsilon)$ is FPT for any $0 < \epsilon < p$, then P = NP*.*

Proof. We present a proof for NP-maximization problems and towards the end we outline the necessary changes needed for this proof to work for minimization problems. Therefore let Q be an NP-maximization problem and suppose that for some $0 < \epsilon < p$, the parameterized problem $Q_a(\epsilon)$ is fixed-parameter tractable. Let $\mathscr{A}$ be an FPT-algorithm for it with run time $O(t(k)\text{poly}(|I|))$. We will use $\mathscr{A}$ to solve the underlying decision problem of Q in polynomial time. Note that for $0 < \epsilon < p$, $g(n) - f(n) - \epsilon n$ is strictly increasing and strictly positive for large enough values of n.

Let (I, s) be an instance of the decision version of Q. Then (I, s) is a YES-instance if and only if $\max(I) \geq s$. We consider three cases and proceed as described below.

Case 1: $s < f(|I|)$.

Since $\max(I) \geq f(|I|)$, we answer YES.

Case 2: $f(|I|) \leq s < f(|I|) + \epsilon|I|$.

In this case, we claim that we can transform the input instance (I, s) into an 'equivalent' instance (I', s') such that

1. $f(|I'|) + \epsilon|I'| \leq s'$.

2. $|I'| = \text{poly}(|I|)$.

3. $\text{opt}(I) \geq s$ if and only if $\text{opt}(I') \geq s'$.

This will show that we can, without loss of generality, go to Case 3 below directly.

To achieve the transformation, add a TUB instance I_1 to I. Define $I' = I + I_1$ and $s' = s + g(|I_1|)$. Then it is easy to see that $\max(I) \geq s$ if and only if $\max(I') \geq s'$. We want to choose I_1 such that $f(|I'|) + \epsilon|I'| \leq s'$. Since $|I'| = |I| + |I_1|$ and $s' = s + g(I_1)$, and since $f(|I|) < s$, it suffices to choose I_1 satisfying

$$f(|I| + |I_1|) + \epsilon|I| + \epsilon|I_1| \leq f(|I|) + g(|I_1|)$$

By Property $P1$, we have $f(|I| + |I_1|) \leq f(|I|) + f(|I_1|) + c^*$, so it suffices to satisfy

$$f(|I_1|) + c^* + \epsilon|I| + \epsilon|I_1| \leq g(|I_1|)$$

By Property P2 we have $g(|I_1|) > f(|I_1|) + p|I_1|$, so it suffices to satisfy

$$c^* + \epsilon|I| \leq (p - \epsilon)|I_1|$$

Such an instance I_1 (of size polynomial in $|I|$) can be chosen because $0 < \epsilon < p$, and because the tight upper bound is polynomial-time uniform dense.

Case 3: $f(|I|) + \epsilon|I| \leq s$

In this case, we transform the instance (I, s) into an instance (I', s') such that

1. $f(|I'|) + \epsilon|I'| + c_1 = s'$, where $0 \leq c_1 \leq c_0$ and c_0 is a fixed constant.
2. $|I'| = \text{poly}(|I|)$.
3. $\max(I') \geq s'$ if and only if $\max(I) \geq s$.

We then run algorithm $\mathscr{A}$ with input (I', c_1). Algorithm $\mathscr{A}$ answers YES if and only if $\max(I') \geq s'$. By condition 3 above, this happens if and only if $\max(I) \geq s$. This takes time $O(t(c_1) \cdot \text{poly}(|I'|))$.

We obtain I' by adding a TLB instance I_1 to I. What if addition of any TLB instance yields an I' with $s' < f(I') + \epsilon|I'|$? In this case, s must already be very close to $f(|I|) + \epsilon|I|$; the difference $k \triangleq s - f(|I|) - \epsilon|I|$ must be at most $\epsilon d + c^*$, where d is the size of the smallest TLB instance I_0. (Why? Add I_0 to I to get $s + f(d) < f(|I| + d) + \epsilon(|I| + d)$; applying property P1, we get $s + f(d) < f(|I|) + f(d) + c^* + \epsilon|I| + \epsilon d$, and so $k < c^* + \epsilon d$.) In such a case, we can use the FPT algorithm $\mathscr{A}$ with input (I, k) directly to answer the question "Is $\max(I) \geq s$?" in time $O(t(\epsilon d + c^*) \cdot \text{poly}(|I|))$.

So now assume that $k \geq c^* + \epsilon d$, and it is possible to add TLB instances to $|I|$. Since f is an increasing function, there is a *largest* TLB instance I_1 we can add to I to get I' while still satisfying $s' \geq f(I') + \epsilon|I'|$. The smallest TLB instance bigger than I_1 has size at most $|I_1| + c$, where c is the constant that appears in the definition of density. We therefore have the following inequalities

$$f(|I'|) + \epsilon|I'| \leq s' < f(|I'| + c) + \epsilon(|I'| + c).$$

Since f is increasing and satisfies property P1, we have

$$\big[f(|I'|+c)+\epsilon(|I'|+c)\big]-\big[f(|I'|)+\epsilon|I'|\big] \leq f(c)+c^*+\epsilon c \triangleq c_0,$$

and hence $s' = f(|I'|)+\epsilon|I'|+c_1$, where $0 \leq c_1 \leq c_0$. Note that c_0 is a constant independent of the input instance (I,s). Also, since Q is a polynomially bounded problem, $|I_1|$ is polynomially bounded in $|I|$.

Note that the proof for Cases 2 and 3 do not make explicit use of the fact that Q is a maximization problem; the proof here goes through for minimization problems as well. In fact, the only change necessary for minimization problems is in Case 1 where if $s <$ TLB, we return NO. □

Remark 2.1. Note that there are some problems, notably MAX 3-SAT, for which the constant c_0 in Case 3 of the proof above, is 0. For such problems, the proof of Theorem 2.6 actually proves that the problem $Q' = \{(I,k) : \max(I) \geq f(|I|)+\epsilon|I|\}$ is NP-hard. But in general, the constant $c_0 \geq 1$ and so this observation cannot be generalized.

The constraints imposed in Theorem 2.6 seem to be rather strict, but they are satisfied by a large number of NP-optimization problems.

Corollary 2.1. *For any* NP*-optimization problem Q in the following list, the $Q_a(\epsilon)$ problem is not fixed-parameter tractable unless* P = NP*:*

Problem	TLB$(I)+\epsilon\cdot\|I\|+k$	*Range of* ϵ
1. MAX SAT	$(\frac{1}{2}+\epsilon)m+k$	$0<\epsilon<\frac{1}{2}$
2. MAX c-SAT	$(\frac{1}{2}+\epsilon)m+k$	$0<\epsilon<\frac{1}{2}$
3. MAX EXACT c-SAT	$(1-\frac{1}{2^c}+\epsilon)m+k$	$0<\epsilon<\frac{1}{2^c}$
4. MAX LIN-2	$(\frac{1}{2}+\epsilon)m+k$	$0<\epsilon<\frac{1}{2}$
5. PLANAR INDEPENDENT SET	$(\frac{1}{4}+\epsilon)n+k$	$0<\epsilon<\frac{3}{4}$
6. INDEPENDENT SET-B	$(\frac{1}{B+1}+\epsilon)n+k$	$0<\epsilon<\frac{B}{B+1}$
7. DOMINATING SET-B	$(\frac{1}{B+1}+\epsilon)n+k$	$0<\epsilon<\frac{B}{B+1}$
8. VERTEX COVER-B	$\frac{m}{B}+\epsilon n+k$	$0<\epsilon<h(B)$
9. MAX ACYCLIC SUBGRAPH	$(\frac{1}{2}+\epsilon)m+k$	$0<\epsilon<\frac{1}{2}$
10. MAX PLANAR SUBGRAPH	$(1+\epsilon)n-1+k$	$0<\epsilon<2$
11. MAX CUT	$\frac{m}{2}+\lceil\frac{n-c}{4}\rceil+\epsilon n+k$	$0<\epsilon<\frac{1}{4}$
12. MAX DICUT	$\frac{m}{4}+\sqrt{\frac{m}{32}+\frac{1}{256}}-\frac{1}{16}+\epsilon m+k$	$0<\epsilon<\frac{3}{4}$

Remark 2.2. Note that for VERTEX COVER-B, the lower bound is expressed in terms of the number of edges m whereas the question asked is whether there exists a cover of size at most TLB $+\epsilon\cdot n+k$. For MAX CUT, the lower bound is in terms of the number of vertices n and edges m but the question asked is whether there exists a cut of size at least TLB $+\epsilon\cdot n+k$. In both these cases, the parameter used to express the input size in the "$\epsilon\cdot f(|I|)$

part" is not the same as that in the expression for the lower bound. Hence these do not satisfy the conditions of Theorem 2.6 but can be proved independently using the same proof-technique. The function $h(B)$ in Item 8 of Corollary 2.1 is given by

$$h(B) = \frac{(B-1)(2B+1)}{2B(B+1)}.$$

We now extend Theorem 2.6 to the corresponding variant of the below-guarantee question. Let $Q = \{\mathscr{I}, S, V, \text{opt}\}$ be an NPO problem with a tight lower bound $f : \mathbb{N} \to \mathbb{N}$ and a tight upper bound $g : \mathbb{N} \to \mathbb{N}$ which are increasing functions and satisfy the following conditions

P3 For all $a, b \in \mathbb{N}$, $g(a+b) \leq g(a) + g(b) + c^*$, where c^* is a constant,

P4 There exists $r \in \mathbb{Q}^+$ such that $g(n) - f(n) > rn$ for all $n \geq n_0$ for some $n_0 \in \mathbb{N}$.

Theorem 2.7. *Let $Q = \{\mathscr{I}, S, V, opt\}$ be a polynomially bounded* NPO *problem such that the following conditions hold.*

1. *Q is partially additive.*
2. *Q has a tight lower bound (*TLB*) f such that the infinite family of instances $\mathscr{I}'$ witnessing the tight lower bound is dense poly-time uniform with respect to the condition $opt(I) = f(|I|)$.*
3. *Q has a tight upper bound (*TUB*) g which is increasing, satisfies condition P3, and with f satisfies P4. The infinite family of instances $\mathscr{I}'$ witnessing the tight upper bound is dense poly-time uniform with respect to the condition $opt(I) = g(|I|)$.*
4. *The underlying decision problem $\tilde{Q}$ of Q is* NP*-hard.*

Let $p := \sup\{r \in \mathbb{Q}^+ : g(n) - f(n) > rn \text{ for all } n \geq n_0\}$ and for $0 < \epsilon < p$, define $Q_b(\epsilon)$ to be the following parameterized problem

$$Q_b(\epsilon) = \{(I,k) : \ max(I) \geq \text{TUB}(I) \ - \epsilon \cdot |I| - k\} \quad \textit{(max problems);}$$
$$Q_b(\epsilon) = \{(I,k) : \ min(I) \leq \text{TUB}(I) \ - \epsilon \cdot |I| - k\} \quad \textit{(min problems).}$$

If $Q_b(\epsilon)$ is FPT for any $0 < \epsilon < p$, then P = NP*.*

Proof Sketch. We sketch a proof for NP-minimization problems. Assume that for some ϵ in the specified range, $Q_b(\epsilon)$ is indeed in FPT. Consider an instance (I, s) of the underlying decision version of Q. Here is a P-time procedure for deciding it. If $s >$ TLB, then the answer is trivially YES. If s lies between TUB and TUB $- \epsilon|I|$, then "add" a gadget of suitable size corresponding to the TLB to obtain an equivalent instance (I', s'). This

increases the input size, but since we are adding a gadget whose optimum value matches the lower bound, the increase in the optimum value of I' is less than proportional, so that now s' is less than $\text{TUB}-\epsilon|I'|$. If s were already less than $\text{TUB}-\epsilon|I|$, then "add" a gadget of suitable size corresponding to the TUB to obtain an equivalent instance (I', s'). This increases the optimum value faster than it does the instance size, so that now s' is less than $\text{TUB} - \epsilon|I'|$ by only a constant, say c_1. Use the hypothesized FPT algorithm for $Q_b(\epsilon)$ with input (I', c_1) to correctly decide the original question. □

The next result shows that for a number of NP-optimization problems, the below-guarantee parameterized variant is unlikely to be in FPT.

Corollary 2.2. *For any* NP*-optimization problem Q in the following list, the $Q_b(\epsilon)$ problem is not fixed-parameter tractable unless* P = NP*:*

Problem	$\text{TUB}(I) - \epsilon \cdot \lvert I\rvert - k$	*Range of* ϵ
1. Max Sat	$(1-\epsilon)m - k$	$0 < \epsilon < \frac{1}{2}$
2. Max c-Sat	$(1-\epsilon)m - k$	$0 < \epsilon < \frac{1}{2}$
3. Max Exact c-Sat	$(1-\epsilon)m - k$	$0 < \epsilon < \frac{1}{2^c}$
4. Max Lin-2	$(1-\epsilon)m - k$	$0 < \epsilon < \frac{1}{2}$
5. Planar Independent Set	$(1-\epsilon)n - k$	$0 < \epsilon < \frac{3}{4}$
6. Independent Set-B	$(1-\epsilon)n - k$	$0 < \epsilon < \frac{B}{B+1}$
7. Dominating Set-B	$(1-\epsilon)n - k$	$0 < \epsilon < \frac{B}{B+1}$
8. Vertex Cover-B	$(\frac{B}{B+1} - \epsilon)n - k$	$0 < \epsilon < \frac{(B-1)(2B+1)}{2B(B+1)}$
9. Max Acyclic Subgraph	$(1-\epsilon)m - k$	$0 < \epsilon < \frac{1}{2}$
10. Max Planar Subgraph	$(3-\epsilon)n - 6 - k$	$0 < \epsilon < 2$
11. Max Cut	$m - \epsilon n - k$	$0 < \epsilon < \frac{1}{4}$
12. Max Dicut	$(1-\epsilon)m - k$	$0 < \epsilon < \frac{3}{4}$

Remark 2.3. For Max Cut, the upper bound is expressed in terms of the number of edges, whereas the parameter used to express the input size in the "$\epsilon \cdot g(|I|)$ part" is the number of vertices. This again does not satisfy the conditions of Theorem 2.7 but can be proved independently using the same proof-technique.

2.7 When the Guarantee is a Structural Parameter

Thus far, we examined the situation when the lower bound is a function of the input size. We now consider problems where the lower bound is a function of the problem instance rather than the input size.

2.7.1 Above-Guarantee Vertex Cover

In Chapter 1 we discussed the standard and above-guarantee parameterizations of Vertex Cover. To recapitulate, the standard parameterized

version (given a graph $G = (V, E)$, does it have a vertex cover of size at most k?) is fixed-parameter tractable by a number of algorithms (see [107]). However for a given instance (G, k) of the problem to be a YES-instance, the standard parameter k must be at least the size of a maximum matching in G. Thus $k = \Omega(|V|)$ in general and any FPT-algorithm for the standard version takes time exponential in $|V|$. It is therefore natural to consider the above-guarantee version of VERTEX COVER.

ABOVE-GUARANTEE VERTEX COVER
Input: A graph G with a maximum matching of size μ.
Parameter: A nonnegative integer k.
Question: Does G have a vertex cover of size at most $\mu + k$?

Note that the lower bound guarantee for this problem is different in the sense that it is a function of the *input instance* and not the *input size*. The ABOVE-GUARANTEE VERTEX COVER is fixed-parameter reducible [102] (in fact, fixed-parameter equivalent [57]) to MIN 2-SAT DELETION. See Chapter 4 for more details.

The parameterized complexity of these problems was open for quite some time until recently Razgon et al. [116] showed the latter problem to be fixed-parameter tractable. This also shows that ABOVE-GUARANTEE VERTEX COVER is fixed-parameter tractable. Interestingly, it was already known that one can check in polynomial time whether the size of a minimum vertex cover equals that of a maximum matching [44]. This is in contrast to the problems considered before (problems in Section 2.4) where we do not know whether there exists a polynomial time algorithm to decide whether a given instance has an optimum value equal to the tight lower bound. For a discussion on this issue see Section 2.8.

2.7.2 The Kemeny Score Problem

The KEMENY SCORE problem is a rank-aggregation problem that arises in social choice theory [49, 15]. Informally, the goal of this problem is to combine a number of different rank orderings on the same set of candidates to obtain a "best" ordering. An election $(\mathcal{V}, C)$ consists of a set $\mathcal{V}$ of votes and set C of candidates. A *vote* is simply a preference list of candidates (a permutation of C). A "Kemeny consensus" is a preference list of candidates that is "closest" to the given set of votes. Given a pair of votes π_1, π_2, the *Kendall-Tau-distance* (*KT-distance* for short) between π_1 and π_2 is defined as

$$\text{dist}(\pi_1, \pi_2) = \sum_{\{c,d\} \subseteq C} d_{\pi_1,\pi_2}(c, d),$$

where $d_{\pi_1,\pi_2}(c, d) = 0$ if π_1 and π_2 rank c and d in the same order, and 1 otherwise. The *score* of a preference list π with respect to an election $(\mathcal{V}, C)$

is defined as

$$\mathrm{scr}(\pi) = \sum_{\pi_i \in \mathcal{V}} \mathrm{dist}(\pi, \pi_i).$$

A preference list π with minimum score is called a *Kemeny consensus* of the election $(\mathcal{V}, C)$ and its score $\mathrm{scr}(\pi)$ is called the *Kemeny score* of $(\mathcal{V}, C)$.

The Kemeny Score problem is the following.

Kemeny Score
Input: An election $(\mathcal{V}, C)$ and an integer k.
Parameter: The integer k.
Question: Is the Kemeny score of $(\mathcal{V}, C)$ at most k?

This problem is NP-complete even for the case when the number of votes is four, whereas the complexity of the case $|\mathcal{V}| = 3$ is still open [49]. The case $|\mathcal{V}| = 2$ can be solved trivially as the Kemeny score is simply the KT-distance of the two votes. In [15], Kemeny Score was shown to be in FPT. Also note that there are at most $|C|!$ distinct votes and therefore for constant $|C|$, the problem is polynomial-time solvable irrespective of the number of votes.

Observe that given an election $(\mathcal{V}, C)$ and a preference list π, the score of π with respect to $(\mathcal{V}, C)$ is lower bounded by

$$L_{\mathcal{V},C} = \sum_{\{a,b\} \in C} \min\{\nu(a, b), \nu(b, a)\},$$

where $\nu(a, b)$ is the number of preference lists in $\mathcal{V}$ that rank a higher than b. Thus the Kemeny score of $(\mathcal{V}, C)$ is lower bounded by $L_{\mathcal{V},C}$. Observe that this lower bound is tight since in the case where $\mathcal{V}$ contains $|\mathcal{V}| - 1$ identical preference lists along with a single copy of the reverse of these lists, the Kemeny score of $(\mathcal{V}, C)$ equals $L_{\mathcal{V},C} = \binom{|C|}{2}$.

Hence a more natural question is:

Above-Guarantee Kemeny Score
Input: An election $(\mathcal{V}, C)$ and an integer k.
Parameter: The integer k.
Question: Is the Kemeny score of $(\mathcal{V}, C)$ at most $L_{\mathcal{V},C} + k$?

We note that this problem is fixed-parameter tractable by a parameter-preserving reduction to a weighted variant of Directed Feedback Vertex Set, where the vertices have weights and one has to decide whether there exists a feedback vertex set of weight at most k, with k as parameter.

The following reduction appears in [49]. Given an election $(\mathcal{V}, C)$, construct an arc-weighted directed graph G on $|C|$ vertices as follows: for each

pair of vertices u and v, there exists an arc from u to v if and only if the majority of the votes in $\mathcal{V}$ rank u higher than v; the weight of the arc from u to v, $w(u,v)$, is the difference between the number of votes that rank u higher than v and vice versa. Note that in case of tie, there are no arcs between u and v.

Lemma 2.3. *The election $(\mathcal{V}, C)$ has a Kemeny score of at most $L_{\mathcal{V},C} + k$ if and only if G has a feedback arc set of weight at most k.*

Proof. Assume that the election $(\mathcal{V}, C)$ has a Kemeny score of at most $L_{\mathcal{V},C}+k$ and let π be a permutation that attains this score. Let S be the set of arcs $(b,a) \in A(G)$ such that π ranks a before b (but the majority of lists in $\mathcal{V}$ rank b before a). Then $k \leq \sum_{(b,a)\in S} w(b,a)$. We claim that if we delete the arcs in S, then the resulting digraph is acyclic. For if $(a_1, a_2, \ldots, a_k, a_{k+1} = a_1)$ is a dicycle in G, then at least one of the arcs (a_i, a_{i+1}) will have been deleted. For otherwise, in the permutation π, we would have a_i preceding a_{i+1} for all $1 \leq i \leq k$, an impossibility.

Conversely, suppose that G has a feedback arc set T of weight at most k. Let π be a topological ordering of the vertices in the DAG obtained by deleting the arc-set T from G. We claim that $\mathrm{scr}(\pi) \leq L_{\mathcal{V},C} + k$. If π ranks a before b then the contribution of the pair $\{a,b\}$ to $\mathrm{scr}(\pi)$ is $\nu(b,a)$. If $(a,b) \in E(G)$ then $\nu(b,a) = \min\{\nu(a,b), \nu(b,a)\}$; if $(b,a) \in E(G)$ then

$$\nu(b,a) = w(b,a) + \nu(a,b) = w(b,a) + \min\{\nu(a,b), \nu(b,a)\}.$$

Note that the arcs in T are precisely those arcs $(b,a) \in E(G)$ such that π ranks a before b and therefore $k = \sum_{\pi(a)<\pi(b)} w(b,a)$. Hence $\mathrm{scr}(\pi) = \sum_{\pi(a)<\pi(b)} \nu(b,a) = L_{\mathcal{V},C} + k$. □

The problem of deciding whether an arc-weighted directed graph has a feedback arc set of size at most k fixed-parameter reduces to the problem of deciding whether a vertex-weighted directed graph has a feedback vertex set of size at most k [53]. This latter problem is in FPT since the algorithm for Directed Feedback Vertex Set presented in [31] actually works for this vertex-weighted variant of the problem. Consequently,

Theorem 2.8. Above-Guarantee Kemeny Score *is in* FPT.

Open Problem 2.1. Are there other "natural" problems where the lower bound is a function of the input instance rather than the input size? If so, are their above guaranteed versions fixed-parameter tractable?

2.8 Conclusion and Further Research

We have argued that for several optimization problems including all those in MAX SNP, the above or below-guarantee parameterization is the natural

and more practical direction to pursue. In Section 2.5 we exhibited two problems for which the above-guarantee parameterization is hard and there are problems such as MAX SAT [98], MIN LINEAR ARRANGEMENT [76] and MIN PROFILE [82, 77] for which this question is fixed-parameter tractable. The main problem left open is:

Open Problem 2.2. Is there a characterization for the class of problems for which the above or below-guarantee question with respect to a tight lower or upper bound is in FPT (or W[1]-hard)?

We believe that there are several natural directions to pursue both from an algorithmic as well as from a practical point of view. As stated before, not many results are known on parameterized above or below-guarantee problems. In fact, the complexity of problems (1) through (7) stated in Section 2.4, when parameterized above their guaranteed values, is open. Some of the more interesting above-guarantee problems are:

Open Problem 2.3. PLANAR INDEPENDENT SET: Given an n-vertex planar graph and an integer parameter k, does G have an independent set of size at least $\lceil n/4 \rceil + k$?

Open Problem 2.4. MAX CUT: Given a connected graph G on n vertices and m edges and a parameter k, does G have a cut of size at least $\lceil m/2 \rceil + \lceil (n-1)/2 \rceil /2 + k$?

Open Problem 2.5. MAX EXACT c-SAT: Given a Boolean CNF formula F with m clauses such that each clause has exactly c distinct literals and an integer parameter k, does there exist an assignment that satisfies at least $(1 - 2^{-c})m + k$ clauses?

In all the above problems, the case $k = 0$ is interesting in itself since an FPT-algorithm for any of the above problems implies a polynomial-time test for deciding whether the optimum equals the lower bound for that particular problem. In fact, one way to prove that an above-guarantee problem *does not* have an FPT-algorithm is by showing that there is no polynomial time algorithm (assuming P $\neq$ NP) that decides whether the optimum equals the guaranteed lower bound.

Open Problem 2.6. Is there a polynomial time algorithm that decides whether

1. a given planar graph G on n vertices has a maximum independent set of size exactly $\lceil n/4 \rceil$?

2. a given connected graph G with m edges and n vertices has a maximum cut of size exactly $\lceil m/2 \rceil + \lceil (n-1)/2 \rceil /2$?

3. a given Boolean CNF formula F with m clauses with exactly c distinct literals per clause, has a maximum of $(1-2^{-c})m$ simultaneously satisfiable clauses?

Here are some interesting below-guarantee problems which are open:

Open Problem 2.7. [102] König Edge Deletion Set: Given a graph G on n vertices, m edges and an integer k, does there exist an edge-induced subgraph of G on $m-k$ edges that is König (a graph in which a minimum vertex cover and a maximum matching have the same size)? In other words, can k edges be deleted from G to make it König? In Chapter 4 we show that the related vertex version is in FPT.

Open Problem 2.8. [36, 105] Perfect Vertex Deletion: Let G be a graph on n vertices and m edges and k a nonnegative integer. Does there exist a vertex-induced subgraph on $n-k$ vertices that is perfect? A similar question can be framed for the edge version.

2.8.1 Approximating the Above-Guarantee Parameter

We examine the notion of an above-guarantee approximation algorithm. The idea, as in recent attempts to study parameterized approximation [27, 48, 33], is to try and approximate the parameter k by an efficient algorithm. To motivate this discussion, we consider the Vertex Cover problem once more. As we observed already, a graph G with a maximum matching of size μ has a minimum vertex cover of size $\beta(G) = \mu + k$ for some $0 \leq k \leq 2\mu$. A 2-approximate algorithm for this problem simply includes all vertices of a maximum matching. What is interesting is that no polynomial time algorithm is known for this problem which has an approximation factor a constant strictly less than 2. In fact, it is now an outstanding open problem whether there indeed exists such a polynomial time approximation algorithm.

An above-guarantee approximation algorithm for the Vertex Cover problem tries to approximate k instead of the "entire" vertex cover. For example, an algorithm which outputs a solution of size at most $\mu+\alpha(\beta(G)-\mu)$, where $\alpha > 1$, performs better than the 2-approximate algorithm (the one which includes all vertices of a maximum matching) whenever $\mu+\alpha(\beta(G)-\mu) < 2\mu$, that is, whenever $\beta(G) - \mu < \mu/\alpha$. Since $\mu = O(n)$, this means that whenever $\beta(G) - \mu < O(n/\alpha)$, the additive approximation algorithm beats the 2-approximate algorithm. Here n is the number of vertices in the input graph.

Formally, one can define an above-guarantee α-approximate algorithm as follows. If Q is an NP-maximization problem with a tight lower bound (tlb) on its optimal solution size, then an above-guarantee α-approximate algorithm for Q takes as input an instance I of Q and outputs a solution of

size at least $\text{TLB}(I) + \alpha(\text{opt}(I) - \text{TLB}(I))$ in time polynomial in $|I|$. For an NP-minimization problem, an α-approximate algorithm outputs a solution of size at most $\text{TLB}(I) + \alpha(\text{opt}(I) - \text{TLB}(I))$ in time polynomial in $|I|$. Note that for an NP-maximization problem $\alpha < 1$; for a minimization problem $\alpha > 1$. One can now think of developing above-guarantee approximate algorithms for other problems which have a guaranteed lower bound on their solution size.

Open Problem 2.9. Does there exist an above-guarantee approximation algorithm for the following problems?

1. Planar Independent Set.
2. Max Cut.

We note that the standard optimization versions of both these problems have good approximation algorithms. The Planar Independent Set problem has a PTAS due to Baker [11]. For Max Cut, there exists a 0.879-approximate algorithm due to Goeman and Williamson [68]. Baker's algorithm takes as input a planar graph G and a positive integer p and outputs an independent set of size at least $p/(p+1)$ times optimal in time $O(8^p p|V(G)|)$. Thus if a maximum independent set has size $n/4 + k$, Baker's algorithm outputs a solution of size at least $p/(p+1)\cdot(n/4+k)$. Here again an above-guarantee approximate algorithm which outputs a solution of size at least $n/4 + \alpha k$, α a constant, yields a better solution whenever

$$\frac{p}{p+1}\left(\frac{n}{4} + k\right) < \frac{n}{4} + \alpha k,$$

that is, whenever

$$k < \frac{n}{4\{p - \alpha(p+1)\}}.$$

Thus if the optimum is only a "small distance" away from the lower bound, an above-guarantee approximation performs better than the PTAS. A similar observation can be made for the Max Cut problem. This could be the motivation for developing such algorithms.

In summary, we believe that parameterizing above or below guaranteed bounds is an interesting paradigm both from a theoretical and practical point of view with several important problems yet to be explored.

Chapter 3

Approximating Beyond the Limit of Approximation

In the last chapter, we saw that if an NP-optimization problem Q admits an α-approximation algorithm, then the $\alpha \cdot \text{OPT} \pm k$ question is in FPT provided the standard parameterized version of Q is in FPT. In this chapter we consider a related question, that of approximating beyond the limit of approximation. The problem that we wish to address is this: Suppose an NP-maximization problem admits a polynomial-time α-approximation algorithm, where $\alpha < 1$. What then is the complexity of finding an $(\alpha+\epsilon)$-approximate solution given an arbitrary ϵ with $0 < \epsilon < 1-\alpha$? Since α could be the limit of approximation for the problem, the algorithm in question may require exponential time but the objective is to find a better-than-α-approximate solution much faster than it would take to compute an exact solution. For minimization problems, we are interested in finding an $(\alpha-\epsilon)$-approximate solution, where $\alpha > 1$ and $0 < \epsilon < \alpha - 1$.

Dantsin et al. [42] answer this question for Max Sat, Max 2-Sat and Max 3-Sat. For Max Sat and Max 2-Sat, they give an algorithm that constructs an $(\alpha+\epsilon)$-approximate solution in time $O^*(\phi^{\epsilon(1-\alpha)^{-1}}m)$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio and m is the number of clauses in the input formula. For Max 3-Sat, they construct an $(\alpha+\epsilon)$-approximate solution in time $O^*(2^{\epsilon(1-\alpha)^{-1}m})$. We show that one can construct an $(\alpha+\epsilon)$-approximate solution for a class of NP-maximization problems (and an $(\alpha-\epsilon)$-approximate solution for NP-minimization problems) "efficiently" provided that there exists an algorithm computing the optimum solution that is "well-behaved" in some sense. We start with some basic definitions regarding approximation algorithms.

3.1 Basic Definitions

Let $Q = \{\mathscr{I}, S, V, \text{opt}\}$ be an NP-optimization problem. For any instance I of Q and any $y \in S(I)$, the approximation ratio $r(I,y)$ of y with respect

to I is defined as

$$r(I, y) := \frac{V(I, y)}{\text{opt}(I)}.$$

Therefore for maximization problems, the approximation ratio is a number ≤ 1; for minimization problems, the ratio is ≥ 1. In either case, the closer the ratio is to 1, the better the solution.

Definition 3.1. Let $Q = \{\mathscr{I}, S, V, \text{opt}\}$ be an NP-optimization problem and let $\alpha > 0$ be a real number.

1. A *polynomial-time α-approximation algorithm* for Q is a polynomial-time algorithm that, given an instance $I \in \mathscr{I}$, computes a solution $y \in S(I)$ such that $r(I, y) \geq \alpha$ (if Q is a maximization problem) and $r(I, y) \leq \alpha$ (if Q is a minimization problem).

2. A *polynomial time approximation scheme (PTAS)* for Q is an algorithm $\mathscr{A}$ that takes as input a pair $(I, k) \in \mathscr{I} \times \mathbb{N}$ such that for every fixed k, the algorithm is a polynomial-time approximation algorithm with ratio $1 + 1/k$ (if Q is a minimization problem) and with ratio $1/(1 + 1/k)$ (if Q is a maximization problem).

Definition 3.2. An NP-optimization problem $Q = \{\mathscr{I}, S, V, \text{opt}\}$ is said to be *polynomially bounded* if there exists a polynomial function $b(\cdot)$ such that for all instances $I \in \mathscr{I}$, we have $\text{opt}(I) \leq b(|I|)$. The function $b(\cdot)$ is called the bound for the problem Q.

Max Sat, for instance, is polynomially bounded since the optimum solution size is bounded above by the number of clauses in the input instance. Graph-theoretic optimization problems such as Vertex Cover, Independent Set, Dominating Set, Feedback Vertex Set, Induced Matching, where the solution is either a vertex or an edge subset, are other examples of polynomially bounded NP-optimization problems. Depending on whether the problem is a vertex or an edge problem, the bound is the size of the vertex or edge set, respectively.

Definition 3.3. An NP-optimization problem Q is a *subset-type optimization problem* if for each instance I of Q there exists a set C_I such that any solution of I is a subset of C_I.

For instance, Max Sat is a subset-type optimization problem since any solution is simply a subset of the set of all clauses of the given input instance. The other graph-theoretic problems mentioned above are subset-type problems too.

3.2 Related Work

The idea of approximating a solution in exponential time is quite recent and only a few papers have appeared on this topic [42, 40, 21]. This idea seems to have originated in the paper by Dantsin et al. [42] where they show how to construct an $(\alpha + \epsilon)$-approximate algorithm for Max Sat that runs in time $O^*(\phi^{\epsilon(1-\alpha)^{-1}m})$, given a polynomial-time α-approximation algorithm. Here $\phi = (1 + \sqrt{5})/2$ is the "golden ratio" and m is the number of clauses in the input formula. The motivation for developing such an algorithm is that while Max Sat admits a 0.758-approximation algorithm [68], it does not have a PTAS. In particular, there exists a constant α such that a polynomial-time α-approximation algorithm for Max Sat implies P = NP [8]. Therefore it is natural to investigate how fast one can obtain an $(\alpha + \epsilon)$-approximate solution albeit in exponential time.

The approach used by Dantsin et al. can be described as follows. Many exponential-time algorithms are recursive and can be viewed as visiting the nodes of an exponential-sized search-tree. The nodes of the search-tree may be viewed as tuples consisting of problem instances and partial solutions. Typically the leaves correspond to either trivial instances or instances solvable in polynomial time. As the algorithm "descends" the search-tree, it builds a solution whose size is typically proportional to the depth of recursion. The idea used by Dantsin et al. is to stop recursing as soon a partial solution of sufficiently large size has been obtained and then applying the approximation algorithm to the resulting instances. In this chapter we show that not only can Max Sat, Max 2-Sat and Max 3-Sat be approximated beyond the approximation lower bound using this technique, but so can any subset-type optimization problem that admits an exponential-time algorithm with certain properties. In particular, we obtain $(\alpha+\epsilon)$-approximate algorithms for Max Sat, Max 2-Sat and Max 3-Sat that run in the same time as those obtained by Dantsin et al. in [42].

Another motivation for considering exponential-time approximation algorithms is that many problems such as Independent Set, Chromatic Number, Subset Sum and a host of graph layout problems such as Bandwidth are hard to approximate in polynomial time (see [9] for more examples). For instance, Håstad [81] showed that Independent Set cannot be approximated in polynomial time to within $n^{1-\epsilon}$ for any $\epsilon > 0$ unless NP = ZPP, where n is the number of vertices in the input graph. Feige and Killian [56] showed the same result for Chromatic Number. Feige [55] also showed that Set Cover cannot be approximated in polynomial time to within a factor $(1-\epsilon)\ln n$, for any $\epsilon > 0$, unless NP $\subseteq$ DTIME$(n^{\log\log n})$, where n is the size of the set to be covered.

The parameterized versions of these problems are either not in FPT unless P = NP or hard for various levels of the W-hierarchy [47]. Faced with such a situation, it is natural to investigate whether one can obtain "good"

(constant-factor) approximation algorithms for problems with running time that is significantly better than the best-known exponential-time algorithms for these problems. This is the approach adopted by Cygan et al. in [40] in which they study two main techniques for developing such algorithms. The first consists of designing a backtracking algorithm with a small search-tree similar to the idea of Dantsin et al. [42]. Using this, they obtain a $(4r-1)$-approximation algorithm for BANDWIDTH in time $O^*(2^{n/r})$, for any $r > 1$.

The second technique is that of "reducing" an exponential-time algorithm to obtain approximate solutions so that the resulting algorithm runs faster than the original algorithm (but still takes exponential time). We illustrate this approach using INDEPENDENT SET as an example. Given a graph $G = (V, E)$, arbitrarily partition the vertex set V into r sets $V_1, \ldots, V_r$ of (almost) equal size. Now find a maximum independent set I_j in each of the graphs $G[V_j]$, $1 \leq j \leq r$, using a brute-force algorithm that runs through all vertex subsets and checks if they are independent. Clearly the largest I_j is an r-approximate solution for G and the time taken to obtain such a set is $O^*(2^{n/r})$, which is better than a running time of $O^*(2^n)$ taken by the algorithm to obtain an exact solution. Cygan et al. refine this idea and obtain a $(1+\ln r)$-approximation algorithm for SET COVER in time $O^*(2^{n/0.31r})$; and an r-approximation algorithm for WEIGHTED DOMINATING SET in time $O^*(2^{0.589n/r})$, for any $r > 1$.

Bourgeois et al. [21] use ideas similar to that of "reductions" described in the previous paragraph to show, among other things, that a problem called MAX HEREDITARY-π can be approximated to any factor $0 < \alpha < 1$ in moderately exponential time. This problem is defined as follows: given a graph $G = (V, E)$ and some hereditary property π, find a subset $V' \subseteq V$ of maximum size such that $G[V']$ satisfies property π. Bourgeois et al. show that given an exact algorithm for MAX HEREDITARY-π with worst-case complexity $O^*(\gamma^n)$, and a rational $0 < \alpha < 1$, there exists an α-approximation algorithm for the problem with running time $O^*(\gamma^{\alpha n})$. Using this result and the analysis of a linear program for the VERTEX COVER problem by Nemhauser and Trotter [106], they show that given an exact algorithm for INDEPENDENT SET with running time $O^*(\gamma^n)$ and a rational $0 < \epsilon < 1$, one can obtain a $(2-\epsilon)$-approximate solution for VERTEX COVER in time $O^*(\gamma^{\epsilon n})$.

Vassilevska et al. [122] study an interesting variation of the approximation versus exact algorithms theme called hybrid algorithms. A *hybrid algorithm* for an NP-optimization problem Q is a collection of algorithms $\mathcal{H} = \{h_1, \ldots, h_p\}$ called *heuristics*, coupled with a polynomial-time procedure S called the *selector*. Given an instance I of Q, the selector returns an index i, $1 \leq i \leq p$, and then the heuristic h_i is executed on I. The purpose of the selector is to select the most appropriate heuristic for a given instance. Vassilevska et al. focus on hybrids consisting of two heuristics: the first is a super-polynomial time exact algorithm and the second is

a polynomial-time approximation algorithm. They go on to show that the Max Cut, Longest Path, Bandwidth and Constraint Satisfaction problems admit hybrid algorithms where a given instance is solved exactly in time better than the best-known exact algorithm for the problem or is approximated in polynomial time to within a ratio exceeding the known inapproximability of the problem.

3.3 Approximating Beyond Approximation Limits

In what follows, we will assume that the NP-optimization problems we consider are subset-type problems and that given an instance I of such a problem, $C(I)$ represents the finite set that contains all possible solutions to I. We also let $\textsc{Opt}(I)$ denote an optimum solution to the instance I (and we let $\text{opt}(I)$ represent the *size* of an optimum solution). We begin by formalizing what we mean by an NP-optimization problem admitting a "well-behaved" exact algorithm.

Definition 3.4. An algorithm $\mathcal{A}$ is an (f, g)-*exact algorithm* for an NP-optimization problem $Q = \{\mathscr{I}, S, V, \text{opt}\}$ if it satisfies the following property: $\mathcal{A}$ takes as instance a tuple $(I, r) \in \mathscr{I} \times \mathbb{N}$ and in at most $f(r, |I|)$ steps

1. either produces an optimum solution if $\text{opt}(I) \leq r$, or
2. produces at most $g(r)$ tuples $(I_1, T_1), \ldots, (I_{g(r)}, T_{g(r)})$, where $T_i \subseteq C(I) \setminus C(I_i)$ and $|T_i| \geq r$, for all $1 \leq i \leq r$, and each I_i is an instance of Q such that for some $1 \leq j \leq g(r)$, $\textsc{Opt}(I_j) \cup T_j = \textsc{Opt}(I)$.

Notice that the conditions $T_i \subseteq C(I) \setminus C(I_i)$ and $\textsc{Opt}(I_j) \cup T_j = \textsc{Opt}(I)$, imply that $\text{opt}(I_j) + |T_j| = \text{opt}(I)$, for some index j. Before proceeding any further let us look at some examples of such exact algorithms. In what follows we let n and m denote the number of vertices and edges, respectively, of a graph, unless otherwise stated.

Example 3.1. Vertex Cover. The following algorithm, that takes a graph $G = (V, E)$ and a positive integer r as input, is an (f, g)-exact algorithm for Vertex Cover, where $f(r, |G|) = 2^{r+1}(n + m)$ and $g(r) = 2^r$. The algorithm picks an arbitrary edge $\{u, v\}$ in the graph that has not yet been covered and considers two cases: either u is in the solution or else v is in the solution. In the first case, the algorithm recurses on the graph $G - u$ after taking u in the solution and in the second it recurses on $G - v$ after taking v in the solution.

The recursion tree may be viewed as a rooted binary tree whose nodes are labelled by pairs of the form (G_i, T_i, l), where G_i is a subgraph of G, $T_i \subseteq V(G)$ and l denotes the depth of node from the root. The root is

labelled $(G, \emptyset, 0)$. The algorithm recurses till depth at most r; if it finds a vertex cover it outputs it, or it outputs the 2^r tuples (G_i, T_i, r) corresponding to the leaves of the recursion tree. Note that partial solutions corresponding to nodes at level i are of size i. It is easy to verify that if $r \geq \text{opt}(G)$ then the algorithm finds an optimal solution; otherwise for some $1 \leq j \leq 2^r$, there exists a node (G_i, T_i, r) such that $\textsc{Opt}(G_i) \cup T_i = \textsc{Opt}(G)$.

The next example that we consider is the Independent Set problem in graphs of bounded degeneracy. A graph is said to be *d-degenerate* if every subgraph of it has a vertex of degree at most d [125]. Examples of graphs of bounded degeneracy include planar graphs (which are 5-degenerate), graphs of bounded degree, graphs of bounded genus, K_h-minor-free graphs, K_h-topological-minor-free graphs, for every fixed positive integer h.

Example 3.2. The Independent Set problem in graphs of degeneracy d. An (f, g)-exact algorithm for this problem with $f(r, |G|) = (d+1)^{r+1}(n+m)$ and $g(r) = (d+1)^r$ is as follows. Given a d-degenerate graph $G = (V, E)$ and a positive integer r, the algorithm picks a vertex u of degree at most d and considers at most $d + 1$ cases. In each case, the algorithm includes exactly one vertex from $N[u]$ in the solution, deletes its closed neighborhood from the graph, and recurses on the new instance thus created. The recursion tree is now a $(d+1)$-ary tree and the algorithm recurses till depth at most r. Again it is easy to see that if $r \geq \text{opt}(G)$ then the algorithm finds an optimal solution, or else outputs the $(d+1)^r$ instance-partial-solution pairs corresponding to the leaves of the tree such that property 2 of Definition 3.4 holds.

Example 3.3. Next consider the Dominating Set problem in graphs of maximum degree at most B, a constant. Given such a graph $G = (V, E)$ and a positive integer r, our algorithm first colors all vertices of G red. It picks a vertex u and considers $B + 1$ cases and, in each case, includes exactly one vertex $v \in N[u]$ in the dominating set, colors all neighbors of v blue, and deletes v from the graph. Intuitively, red vertices are those that are yet to be dominated and the blue ones are those that have already been dominated. If a blue vertex has only blue neighbors then it is deleted from the graph as it plays no role in dominating red vertices. The algorithm recurses till depth r and either ends up with a graph with only blue vertices or with $(B + 1)^r$ instance-partial-solution pairs satisfying property 2 of Definition 3.4. The time taken is $(B + 1)^{r+1}(n + m)$.

Example 3.4. We also note that Mahajan and Raman [98] describe an (f, g)-exact algorithm for Max Sat which takes a Boolean Cnf formula F and a positive integer r as input, and in time

$$f(r, |F|) = O\left(r^2 \cdot \phi^r + |F|\right)$$

either outputs an optimum solution or $g(r) = \phi^r$ instance-partial-solution pairs such that property 2 of Definition 3.4 holds [98]. Here $\phi = (1+\sqrt{5})/2$.

Our first observation regarding (f, g)-exact algorithms is that if the function f is "FPT-like" then the standard parameterized version of the problem in question is fixed-parameter tractable.

Theorem 3.1. *If an* NP*-optimization problem Q admits an (f, g)-exact algorithm such that $f(r, n) = h(r) \cdot p(n)$, where h is a function of r alone and p is a polynomial, then the standard parameterized version of Q is fixed-parameter tractable.*

Proof. Suppose Q is a maximization problem and let $\mathcal{A}$ be an (f, g)-exact algorithm for it. Given an instance (I, k) of the standard parameterized version of Q, run algorithm $\mathcal{A}$ on I for at most $f(k-1, |I|)$ steps. If it outputs a solution then it must be that $\text{opt}(I) \leq k - 1$ and we answer NO. Else answer YES. The case when Q is a minimization problem can be similarly dealt with. □

We now come to the main result of this chapter.

Theorem 3.2. *Let Q be a polynomially-bounded* NP*-maximization problem with bound $b(\cdot)$ that satisfies the following properties:*

1. *Q admits an α-approximation algorithm that runs in time $p(n)$, for some $0 < \alpha < 1$ and some polynomial p;*
2. *Q admits an (f, g)-exact algorithm.*

Then given $0 < \epsilon < 1 - \alpha$ and an instance I of Q of size n, one can compute an $(\alpha + \epsilon)$-approximate solution in time

$$f\left(\frac{\epsilon \cdot b(n)}{1-\alpha}, n\right) + g\left(\frac{\epsilon \cdot b(n)}{1-\alpha}\right) \cdot p(n).$$

Proof. Let $\mathcal{A}$ be an (f, g)-exact algorithm for Q. Given an instance I of size n, run $\mathcal{A}$ on it for $f(c_{\epsilon,n}, n)$ steps, where

$$c_{\epsilon,n} = \frac{\epsilon \cdot b(n)}{1-\alpha}.$$

If the algorithm outputs an optimum solution within this time, then this solution is trivially $(\alpha + \epsilon)$-approximate. Otherwise it outputs $g(c_{\epsilon,n})$ tuples (I_i, T_i). Next run the polynomial-time α-approximation algorithm for Q on the instances I_i, $1 \leq i \leq g(c_{\epsilon,n})$, and let A_i represent the solution thus obtained for instance I_i. Let i be an index for which $|A_i \cup T_i|$ is maximized. We claim that $A_i \cup T_i$ is an $(\alpha + \epsilon)$-approximate solution for I. By the definition of an (f, g)-exact algorithm, there exists $1 \leq j \leq g(c_{\epsilon,n})$

such that $\text{opt}(I) = |T_j| + \text{opt}(I_j)$. Also $|T_j| \geq c_{\epsilon,n}$ and $|A_j| \geq \alpha \cdot \text{opt}(I_j)$. Therefore,

$$\begin{aligned} |A_j| + |T_j| &\geq \alpha \cdot \text{opt}(I_j) + |T_j| \\ &\geq \alpha \cdot \text{opt}(I) + (1-\alpha) \cdot |T_j| \\ &\geq \alpha \cdot \text{opt}(I) + \epsilon \cdot b(n) \\ &\geq (\alpha + \epsilon) \cdot \text{opt}(I). \end{aligned}$$

It is easy to see that the running time is as claimed in the statement of the theorem. □

Since Max Sat has an (f, g)-exact algorithm with $f(r, m) = O(r^2\phi^r + m)$ and $g(r) = \phi^r$, where $\phi = (1 + \sqrt{5})/2$, by Theorem 3.2 we have

Corollary 3.1. *Assume that there exists an α-approximation algorithm for* Max Sat *that runs in time $p(m)$, where m is the number of clauses in the input formula F. Then for any $0 < \epsilon < 1 - \alpha$, one can construct an $(\alpha + \epsilon)$-approximate solution for* Max Sat *in time $O((c^2_{\epsilon,m} + p(m))\phi^{c_{\epsilon,m}})$, where $c_{\epsilon,m} = \epsilon \cdot (1-\alpha)^{-1} \cdot m$ and $\phi = (1 + \sqrt{5})/2$.*

The running time bound obtained in Corollary 3.1 is the same as that obtained by Dantsin et al. in [42].

By Example 3.2, the Independent Set problem in d-degenerate graphs has an (f, g)-exact algorithm where $f(r, |G|) = (d+1)^{r+1} \cdot (n+m)$ and $g(r) = (d+1)^r$. By Theorem 3.2 we have

Corollary 3.2. *If the* Independent Set *problem in d-degenerate graphs admits an α-approximation algorithm that runs in time $p(n)$, where n is the number of vertices in the input graph, then given any $0 < \epsilon < 1 - \alpha$, one can construct an $(\alpha + \epsilon)$-approximate solution in time $O((d+1)^{c_{\epsilon,n}} \cdot p(n))$, where $c_{\epsilon,n} = \epsilon n/(1-\alpha)$.*

For minimization problems, we have the following theorem.

Theorem 3.3. *Let Q be a polynomially-bounded* NP*-minimization problem with bound $b(\cdot)$ that satisfies the following properties:*

1. *Q admits an α-approximation algorithm that runs in time $p(n)$, for some $\alpha > 1$ and some polynomial p;*
2. *Q admits an (f, g)-exact algorithm.*

Then given $0 < \epsilon < \alpha - 1$ and an instance I of Q of size n, one can compute an $(\alpha - \epsilon)$-approximate solution in time

$$f\left(\frac{\epsilon \cdot b(n)}{\alpha - 1}, n\right) + g\left(\frac{\epsilon \cdot b(n)}{\alpha - 1}\right) \cdot p(n).$$

Proof. This proof of this theorem is similar to the one for Theorem 3.2 but we include it for the sake of completeness. Let $\mathcal{A}$ be an (f, g)-exact algorithm for Q. Given an instance I of size n, run $\mathcal{A}$ on I for $f(c_{\epsilon,n}, n)$ steps, where

$$c_{\epsilon,n} = \frac{\epsilon \cdot b(n)}{\alpha - 1}.$$

If the algorithm outputs an optimum solution then this is trivially an $(\alpha-\epsilon)$-approximate solution. Otherwise $\mathcal{A}$ outputs $g(c_{\epsilon,n})$ tuples (I_i, T_i). Next run the polynomial-time α-approximation algorithm for Q on the instances I_i, $1 \leq i \leq g(c_{\epsilon,n})$, and let A_i represent the solution obtained by running this algorithm on I_i. Let i be an index for which $|A_i \cup T_i|$ is maximized. We claim that $A_i \cup T_i$ is an $(\alpha - \epsilon)$-approximate solution for the instance I. By the definition of an (f, g)-exact algorithm, there exists $1 \leq j \leq g(c_{\epsilon,n})$ such that $\text{opt}(I) = |T_j| + \text{opt}(I_j)$. Also $|T_j| \geq c_{\epsilon,n}$ and $|A_j| \leq \alpha \cdot \text{opt}(I_j)$. Therefore,

$$\begin{aligned} |A_j| + |T_j| &\leq \alpha \cdot \text{opt}(I_j) + |T_j| \\ &\leq \alpha \cdot \text{opt}(I) - (\alpha - 1) \cdot |T_j| \\ &\leq \alpha \cdot \text{opt}(I) - \epsilon \cdot b(n) \\ &\leq (\alpha - \epsilon) \cdot \text{opt}(I). \end{aligned}$$

It is easy to see that the running time is as claimed in the statement of the theorem. □

Since Vertex Cover admits a 2-approximation algorithm that runs in time $O(m)$ [38] and an (f, g)-exact algorithm with $f(r, |G|) = 2^{r+1}(n + m)$ and $g(r) = 2^r$, we have

Corollary 3.3. *For every $0 < \epsilon < 1$, there exists an approximation algorithm for the* Vertex Cover *problem with ratio $2 - \epsilon$ that runs in time $O(2^{\epsilon n} \cdot m)$.*

Note that this running time is worse than that obtained by Bourgeois et al. [21] for a $(2 - \epsilon)$-approximation algorithm for Vertex Cover, but our results are more general and have wider applicability.

For the Dominating Set problem in graphs with degree bounded by B, there exists an (f, g)-exact algorithm with $f(r, |G|) = (B + 1)^{r+1}(n + m)$ and $g(r) = (B + 1)^r$.

Corollary 3.4. *Suppose that the* Dominating Set *problem in graphs with degree bounded by B, admits an α-approximation algorithm that runs in time $p(n)$. Then given any $0 < \epsilon < \alpha - 1$, there exists an approximation algorithm for this problem with ratio $\alpha - \epsilon$ that runs in time $O((B + 1)^{c_{\epsilon,n}} \cdot p(n))$, where $c_{\epsilon,n} = \epsilon n/(\alpha - 1)$.*

We mention one final result of this type. The Almost d-Regular Induced Subgraph problem is defined as follows: given a graph G, what is the fewest number of vertices that need to be deleted so that the resulting graph has maximum degree d? In [103], Moser and Thilikos give an (f, g)-exact algorithm for this problem with $f(r, |G|) = (d+1)^{r+1}(n+m)$ and $g(r) = (d+1)^r$. We therefore obtain

Corollary 3.5. *If the* Almost d-Regular Induced Subgraph *problem admits an α-approximation algorithm that runs in time $p(n)$, then given any $0 < \epsilon < \alpha - 1$, there exists an approximation algorithm for this problem with ratio $\alpha - \epsilon$ that runs in time $O((d+1)^{c_{\epsilon,n}} \cdot p(n))$, where $c_{\epsilon,n} = \epsilon n/(\alpha - 1)$.*

3.4 Concluding Remarks

In this chapter we considered designing (exponential-time) algorithms with an approximation ratio that is a fraction ϵ away from a ratio that can be achieved in practice. For small enough ϵ, these algorithms take moderately exponential time and are much faster than exponential-time algorithms that compute optimum solutions. This line of research is still in its infancy and there are quite a few open questions.

A natural question is whether one can obtain results similar to Theorems 3.2 and 3.3 by relaxing the notion of an (f, g)-exact algorithm. Are there any connections with parameterized enumerability? Another question is to design algorithms for specific problems such as Feedback Vertex Set. This problem admits a 2-approximation algorithm due to Bafna et al. [10] and admits an $O^*(1.755^n)$ exact algorithm due to Gaspers et al. [60]. How quickly can one obtain a $(2-\epsilon)$-approximate solution? What about Directed Feedback Vertex Set which admits an $O(\log n \log\log n)$-ratio algorithm [53]?

Another interesting problem is Max Cut which admits exact algorithms with running times $O^*(2^{\omega n/3})$ [126], $O^*(2^{m/4})$ [54], and $O^*(2^{m/5})$ [92], where n and m denote the number of vertices and edges of the input graph, respectively. This problem also admits a 0.879-approximation algorithm due to the seminal work by Goemans and Williamson [68]. Vassilevska et al. [122] give a hybrid algorithm for Max Cut that given an $\epsilon > 0$, either computes an exact solution in time $2^{\epsilon m}$ or finds a $(1/2 + \epsilon/4)$-approximate solution in linear time. But how fast can one obtain an $(0.879+\epsilon)$-approximate solution for Max Cut?

In summary, we believe that the issues considered in this chapter deserve to be explored further and that there are many interesting questions worth investigating.

Chapter 4

Kőnig Subgraph Problems and Above-Guarantee Vertex Cover

In this chapter we study the complexity of a set of problems which we call Kőnig Subgraph Problems. These problems consist of finding subgraphs of a given graph with the property that the size of a minimum vertex cover equals that of a maximum matching in the subgraph. Graphs that satisfy this property are called Kőnig-Egerváry graphs. More specifically, we look at the following two sets of problems. Given a graph G and a nonnegative integer k,

1. does there exist k vertices (edges) whose deletion makes the graph Kőnig-Egerváry?

2. does there exist k vertices (edges) that induce a Kőnig-Egerváry subgraph?

We show that these problems are NP-complete and study their complexity from the points of view of approximation and parameterized complexity. While this may seem to be a marked departure from the study of parameterizing problems from their default values, we will see that the vertex deletion version is closely related to Above Guarantee Vertex Cover (see Chapters 1 and 2). In our study of the parameterized complexity of the vertex deletion version, we uncover a number of interesting structural results on matchings and vertex covers which may be independent interest.

4.1 History and Motivation

The classical notions of *matchings* and *vertex covers* have been at the center of serious study for several decades in the area of Combinatorial Optimization [97]. In 1931, Kőnig and Egerváry independently proved a result of fundamental importance: in a bipartite graph the size of a maximum matching equals that of a minimum vertex cover [97]. This led to a polynomial-time

algorithm for finding a minimum vertex cover in bipartite graphs. In fact, a maximum matching can be used to obtain a 2-approximation algorithm for the Minimum Vertex Cover problem in general graphs, which is still the best-known constant-factor approximation algorithm for this problem [87]. Interestingly, this min-max relationship holds for a larger class of graphs known as Kőnig-Egerváry graphs and it includes bipartite graphs as a proper subclass. Kőnig-Egerváry graphs will henceforth be called Kőnig graphs.

Kőnig graphs have been studied for a fairly long time from a structural point of view [22, 44, 50, 96, 121]. Both Deming [44] and Sterboul [121] gave independent characterizations of Kőnig graphs and showed that Kőnig graphs can be recognized in polynomial time. Lovász [96] used the theory of matching-covered graphs to give an excluded-subgraph characterization of Kőnig graphs that contain a perfect matching. Korach et al. [50] generalized this and gave an excluded-subgraph characterization for the class of all Kőnig graphs.

A natural optimization problem associated with a graph class $\mathcal{G}$ is the following: given a graph G, what is the minimum number of vertices to be deleted from G to obtain a graph in $\mathcal{G}$? For example, when $\mathcal{G}$ is the class of empty graphs, forests or bipartite graphs, the corresponding problems are Vertex Cover, Feedback Vertex Set and Odd Cycle Transversal, respectively. We call the vertex-deletion problem corresponding to the class of Kőnig graphs the Kőnig Vertex Deletion problem. A set of vertices whose deletion makes a given graph Kőnig is called a Kőnig vertex deletion set. In the parameterized setting, the parameter for vertex-deletion problems is the solution size, that is, the number of vertices to be deleted so that the resulting graph belongs to the given graph class.

In this chapter we define various problems related to finding Kőnig-Egerváry subgraphs and study their complexity from the points of view of parameterized complexity and approximation algorithms. More precisely the problems that we study in this chapter are:

1. Kőnig Vertex (Edge) Deletion (KVD/KED). Given a graph G and a nonnegative integer k, do there exist at most k vertices (respectively, edges) whose deletion results in a Kőnig subgraph?

2. Vertex (Edge) Induced Kőnig Subgraph (VKS/EKS). Given a graph G and a nonnegative integer k, do there exist at least k vertices (respectively, edges) which induce[1] a Kőnig subgraph?

The KVD and VKS problems (and similarly, KED and EKS) are equivalent from the point of view of NP-completeness but differ in their approximability and parameterized complexity.

[1]If E' is an edge-subset, the graph $G[E']$ induced by E' is defined as one with vertex set $V(E')$ and edge-set E'.

While studying the Kőnig Vertex Deletion problem, we first establish interesting structural connections between minimum vertex covers, maximum matchings and minimum Kőnig vertex deletion sets. Using these, we show that Kőnig Vertex Deletion is fixed-parameter tractable when parameterized by the solution size. Note that Kőnig graphs are not hereditary, that is, not closed under taking induced subgraphs. For instance, a 3-cycle is not Kőnig but attaching an edge to one of the vertices of the 3-cycle results in a Kőnig graph. In fact, Kőnig Vertex Deletion is one of the few vertex-deletion problems associated with a non-hereditary graph class whose parameterized complexity has been studied. Another such example can be found in [103].

One motivation for studying Kőnig subgraph problems is that the versions of Kőnig subgraph problems when the resulting graph we look for is bipartite (i.e. replace Kőnig in the above problem definitions by *bipartite*) are well studied in the area of approximation algorithms and parameterized complexity [70, 117, 120]. Kőnig subgraph problems are natural generalizations of bipartite subgraph problems but have not been studied algorithmically. We believe that this can trigger explorations of other questions in Kőnig graphs. Another motivation for studying Kőnig subgraph problems is that Kőnig Vertex Deletion is closely related to Above Guarantee Vertex Cover. As mentioned in Chapter 2, the parameterized complexity of Above Guarantee Vertex Cover was open for quite some time and is now known to be fixed-parameter tractable as we show in this chapter.

The rest of this chapter is organized as follows. In Section 4.2, we describe our notation and state some known results about Kőnig graphs. In Section 4.3 we show that Above Guarantee Vertex Cover is fixed-parameter tractable and discuss its approximability. We next study the parameterized complexity and approximability of Kőnig Vertex Deletion (Section 4.4) and the vertex and edge versions of the Induced Kőnig Subgraph problem (Section 4.5). We conclude in Section 4.6 with a list of open problems. Missing from our list is Kőnig Edge Deletion the parameterized complexity of which is open.

4.2 Preliminaries

In this section we fix our notation and describe some well-known properties of Kőnig graphs.

4.2.1 Notation

Given a graph G, we use $\mu(G)$, $\beta(G)$ and $\kappa(G)$ to denote, respectively, the size of a maximum matching, a minimum vertex cover and a minimum Kőnig vertex deletion set of G. We sometimes use $\tau(G)$ to denote the difference $\beta(G) - \mu(G)$. When the graph being referred to is clear from the con-

text, we simply use μ, β, κ and τ. Given a graph $G = (V, E)$ and two disjoint vertex subsets V_1, V_2 of V, we let (V_1, V_2) denote the bipartite graph with vertex set $V_1 \cup V_2$ and edge set $\{\{u, v\} \colon \{u, v\} \in E \text{ and } u \in V_1, v \in V_2\}$. If B is a bipartite graph with vertex partition $L \uplus R$ then we let $\mu(L, R)$ denote the size of the maximum matching of B. If M is matching and $\{u, v\} \in M$ then we say that *u is the partner of v in M*. If the matching being referred to is clear from the context we simply say *u is a partner of v*. The vertices of G that are the endpoints of edges in the matching M are said to be *saturated by M*; all other vertices are *unsaturated by M*.

4.2.2 Properties of Kőnig Graphs

A graph $G = (V, E)$ is said to be *Kőnig* if $\beta(G) = \mu(G)$. The following lemma follows directly from the definition of Kőnig graphs.

Lemma 4.1. [44, 121] *A graph $G = (V, E)$ is Kőnig if and only if for every bipartition of V into $V_1 \uplus V_2$, with V_1 a minimum vertex cover of G, there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1.*

In order to show that a graph is Kőnig it is actually sufficient to demonstrate the existence of just one bipartition of V into $V_1 \uplus V_2$, with V_1 a vertex cover of G such that there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1.

Lemma 4.2. *A graph $G = (V, E)$ is Kőnig if and only if there exists a bipartition of V into $V_1 \uplus V_2$, with V_1 a vertex cover of G such that there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1.*

Proof. If G is Kőnig then, by Lemma 4.1, there exists a bipartition $V_1 \uplus V_2$, with V_1 a minimum vertex cover of G, such that there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1. Conversely suppose that the vertex set of G can be partitioned as $V_1 \uplus V_2$ such that V_1 is a vertex cover and there exists a matching M across the cut (V_1, V_2) saturating every vertex of V_1. We claim that in fact V_1 is a minimum vertex cover and that M is a maximum matching of G. Suppose that M' is a maximum matching of G and $|M'| > |M|$. Since V_1 is a vertex cover, it picks up at least one endpoint from each edge of M'. Therefore $|V_1| = |M| \geq |M'|$, a contradiction. Therefore M is indeed a maximum matching of G and since any vertex cover of G has size at least $|M|$, it follows that V_1 is a minimum vertex cover of G. □

Lemma 4.3. [44] *Given a graph G on n vertices and m edges and a maximum matching of G, one can test whether G is Kőnig in time $O(n + m)$. If G is indeed Kőnig then one can find a minimum vertex cover of G in this time.*

Since a maximum matching can be obtained in time $O(m\sqrt{n})$ [123], we have

Lemma 4.4. *Let G be a graph on n vertices and m edges. One can check in time $O(m\sqrt{n})$ whether G is Kőnig and, if Kőnig, find a bipartition of $V(G)$ into $V_1 \uplus V_2$ with V_1 a minimum vertex cover of G such that there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1.*

4.3 The Above Guarantee Vertex Cover Problem

In this section we show that Above Guarantee Vertex Cover (AGVC) is fixed-parameter tractable and discuss its approximability. This problem plays a central role in this chapter and the results established here are used in studying the parameterized complexity and approximability of other Kőnig subgraph problems. We will show later (Theorem 4.7) that for the class of graphs with a perfect matching, the parameterized complexity of AGVC and Kőnig Vertex Deletion is the same. For graphs G that do not have a perfect matching, there is a close relationship between $\tau(G)$ and $\kappa(G)$.

Given a graph G it is clear that $\beta(G) \geq \mu(G)$. Recall the definition of AGVC: given a graph G and a nonnegative integer parameter k decide whether $\beta(G) \leq \mu(G) + k$. We first show that for the parameterized complexity of the AGVC problem we may, without loss of generality, assume that the input graph has a perfect matching.

Let $G = (V, E)$ be an undirected graph and let M be a maximum matching of G. Construct $G' = (V', E')$ as follows. Define

$$\begin{aligned} I &= V \setminus V[M] \\ V' &= V \cup \{u' \colon u \in I\} \\ E' &= E \cup \{\{u', v\} \colon \{u, v\} \in E\} \cup \{\{u, u'\} \colon u \in I\}. \end{aligned}$$

Then $M' = M \cup \{\{u, u'\} \colon u \in I\}$ is a perfect matching for G'. Note that $|V(G')| \leq 2|V(G)|$ and $|E(G')| \leq 2|E(G)|$.

Theorem 4.1. *Let G be a graph without a perfect matching and let G' be the graph obtained by the above construction. Then G has a vertex cover of size $\mu(G) + k$ if and only if G' has a vertex cover of size $\mu(G') + k$.*

Proof. Let M denote a maximum matching of G, I denote the set $V(G) \setminus V[M]$ and I' denote the new set of vertices that are added in constructing G'. Clearly, $\mu(G') = \mu(G) + |I|$.

($\Rightarrow$) Let C be a vertex cover of G of size $\mu(G)+k$. Define $C' = C \cup I'$. It is easy to see that C' covers all the edges of G'. Also, $|C'| = \mu(G)+k+|I'| = \mu(G') + k$.

($\Leftarrow$) Let C' be a vertex cover of G' of size $\mu(G')+k$. Define M' to be the set of edges of the form $\{\{u, u'\} \colon u \in I \text{ and } u' \in I'\}$ such that both endpoints

are in C'. One can show that $C = (C' \cap V[M]) \cup \{u \in I : \{u, u'\} \in M'\}$ is a vertex cover of G of size $\mu(G) + k$. □

4.3.1 Parameterized Complexity

We show that AGVC is fixed-parameter tractable by exhibiting a fixed-parameter reduction from AGVC to a problem know as Min 2-Cnf Sat Del [98]. This problem is defined as follows: given a 2-Cnf formula and a nonnegative integer k, do there exist at most k clauses whose deletion makes the resulting formula satisfiable? This problem is NP-complete and its parameterized complexity was open for quite some time. Recently Razgon and O'Sullivan have shown this problem to be fixed-parameter tractable [116].

Theorem 4.2. [116] *Given a* 2-Cnf Sat *formula F on n variables and m clauses and a nonnegative integer k, one can decide whether F has at most k clauses whose deletion makes it satisfiable in time $O(15^k \cdot k \cdot m^3)$. That is, the* Min 2-Cnf Sat Del *problem is fixed-parameter tractable with respect to parameter k.*

We now describe the reduction from AGVC to Min 2-Cnf Sat Del (see [29]). Let $G = (V, E)$ be a graph with a perfect matching P. For every vertex $u \in V$, define x_u to be a Boolean variable. Let $\mathcal{F}(G, P)$ denote the Boolean formula

$$\mathcal{F}(G, P) = \bigwedge_{(u,v)\in P} (\bar{x}_u \vee \bar{x}_v) \bigwedge_{(u,v)\in E} (x_u \vee x_v).$$

Note that $\mathcal{F}(G, P)$ is a formula on $|V|$ variables and at most $2|E|$ clauses.

The proof of the next lemma follows from that of Theorem 5.1 in [29].

Lemma 4.5. *Let $G = (V, E)$ be an n-vertex graph with a perfect matching P. Then G has a vertex cover of size at most $n/2 + k$ if and only if there exists an assignment that satisfies all but at most k clauses of $\mathcal{F}(G, P)$.*

From the proof of Theorem 5.1 in [29], it also follows that given an assignment that satisfies all but at most k clauses of $\mathcal{F}(G, P)$ one can find (in polynomial time) an assignment that satisfies all but at most k clauses of the form $(\bar{x}_u \vee \bar{x}_v)$, where $(u, v) \in P$, that is, clauses that correspond to the perfect matching.

Since Min 2-Cnf Sat Del can be solved in time $O(15^k \cdot k \cdot m^3)$, where m is the number of clauses in the input formula, we have

Theorem 4.3. *Given a graph $G = (V, E)$ and a nonnegative integer parameter k, one can decide whether $\beta(G) \leq \mu(G) + k$ in time $O(15^k \cdot k \cdot |E|^3)$. Moreover if G has a vertex cover of size $\mu(G) + k$ then one can find a vertex cover of this size within this time.*

4.3.2 An Approximation Algorithm

The parameterized version of AGVC asks whether $\tau(G) \leq k$. The optimization version of AGVC is the problem of finding the minimum value of $\tau(G)$. Therefore an approximation algorithm for AGVC approximates the "above-guarantee parameter" rather than the entire vertex cover.

Klein et al. [90] have shown that Min 2-Cnf Sat Del admits a factor-$O(\log n \log\log n)$ approximation algorithm, where n is the number of variables in the 2-Sat formula.

Lemma 4.6. [1, 90] *Given an instance F of* Min 2-Cnf Sat Del*, one can obtain a solution for F in polynomial time that is $O(\log n \log\log n)$ times the optimal solution size, where n is the number of variables in F. If we are willing to allow randomness, we can obtain a solution that is $O(\sqrt{\log n})$ times an optimal solution size.*

We use this algorithm and the reduction from AGVC to Min 2-Cnf Sat Del to obtain an $O(\log n \log\log n)$-approximation algorithm for $\tau(G)$.

Here is an outline of our approximation algorithm: Given a graph G, first apply the construction described before Theorem 4.1, if necessary, to obtain a graph H with a perfect matching P. Note that $\tau(G) = \tau(H)$. Let $\mathcal{F}(H, P)$ denote the 2-Cnf Sat formula obtained from H and P by the construction outlined before Lemma 4.5. Given an assignment that satisfies all but at most k clauses of $\mathcal{F}(H, P)$ one can construct an assignment in polynomial time that satisfies all but at most k clauses of the form $(\bar{x}_u \vee \bar{x}_v)$, where $(u, v) \in P$.

Next use an $O(\log n \log\log n)$ approximation algorithm for Min 2-Cnf Sat Del which "corresponds" to a set $\mathcal{S}$ of edges of the perfect matching P. The set $V(\mathcal{S})$ represents the vertex cover in excess of the matching size and in the graph $G \setminus V(\mathcal{S})$, the sizes of a minimum vertex cover and maximum matching coincide. That is, $G \setminus V(\mathcal{S})$ is Kőnig and therefore by Lemma 4.4 one can obtain a minimum vertex cover C of this graph in polynomial time. Using C and $\mathcal{S}$, reconstruct a vertex cover for G of the appropriate size. The algorithm is presented in Figure 4.1.

Theorem 4.4. *Given an input graph G on n vertices, algorithm **AGVC** finds a vertex cover of G of size $\mu + O(\log n \log\log n)(\beta - \mu)$.*

Proof. The proof follows from the fact that the reduction from AGVC to Min 2-Cnf Sat Del is cost-preserving and that there exists a factor-$O(\log n \log\log n)$ approximation algorithm for the latter. □

Thus this algorithm approximates the deficit between the sizes of a minimum vertex cover and a maximum matching. We mentioned that there exists a 2-approximation algorithm for the Vertex Cover problem and that it is a long-standing open problem to devise a polynomial-time algorithm which has a constant approximation factor less than 2.

AGVC
Input: A graph $G = (V, E)$.
Output: A vertex cover of G of size at most $\mu + O(\log |V| \log \log |V|)(\beta - \mu)$.

1. If G does not have a perfect matching, construct G' as in Theorem 4.1 and set $H \leftarrow G'$; else set $H \leftarrow G$.
2. Find a perfect matching P of H and construct $\mathcal{F}(H, P)$. If G did not have a perfect matching then P is the perfect matching obtained from some maximum matching M of G as described in the construction before Theorem 4.1.
3. Use the approximation algorithm for Min 2-Cnf Sat Del to obtain an $O(\log n \log \log n)$-approximate solution $\mathcal{S}$ for $\mathcal{F}(H, P)$, where $n = |V(H)|$.
4. Obtain a minimum vertex cover C of the Kőnig graph $H \setminus V(\mathcal{S})$, where $V(\mathcal{S})$ is the set of vertices of H corresponding to $\mathcal{S}$.
5. If $H = G$ then return $C \cup V(\mathcal{S})$; else return $(V(\mathcal{S}) \cap V(G)) \cup (V(M) \cap C)$.

Figure 4.1: An approximation algorithm for AGVC.

Our algorithm is better than *any* constant factor approximation algorithm for Vertex Cover whenever

$$\beta - \mu = o\left(\frac{n}{\log n \log \log n}\right)$$

and $\mu = \Omega(n)$. To see this, note that a c-approximate algorithm, $c > 1$, outputs a solution of size $\mu c + (\beta - \mu)c$ whereas our algorithm outputs a solution of size $\mu + O(\alpha(\beta - \mu))$, where $\alpha = \log n \log \log n$. Now if $\beta - \mu = o(n/\alpha)$ and if $\mu = \Omega(n)$, then our algorithm outputs a solution of size $\mu + o(\mu)$, which is better than

$$\beta c = \mu + \mu(c - 1) + (\beta - \mu)c \geq \mu + \Omega(\mu).$$

A randomized approximation algorithm for τ can be obtained by using the $O(\sqrt{\log n})$-randomized approximation algorithm for Min 2-Cnf Sat Del [1], mentioned in Lemma 4.6, in Step 3 of the algorithm.

Theorem 4.5. *There exists a randomized polynomial-time algorithm that takes as input a graph G on n vertices and finds a vertex cover of G of size $\mu + O(\sqrt{\log n})(\beta - \mu)$.*

4.3.3 Hardness of Approximation

We now show that Above Guarantee Vertex Cover and Min 2-Cnf Sat Del do not admit constant-factor approximation algorithms if the Unique Games Conjecture (UGC) [85] is true[2]. In what follows, we

[2]I thank Sundar Vishwanathan for discussions on the results in this section, and in particular, Theorem 4.6.

use the abbreviation VC-PM for the Vertex Cover problem on graphs with a perfect matching.

We make use of the following two results:

Lemma 4.7. [87] *If UGC is true then* Vertex Cover *cannot be approximated to within a factor of* $2-\epsilon$*, for any constant* $\epsilon > 0$.

Lemma 4.8. [29, 124] *If there exists a* $(2-\epsilon)$*-approximation algorithm for* VC-PM *then there exists a* $(2-\epsilon/2)$*-approximation algorithm for* Vertex Cover.

We can now prove the following.

Theorem 4.6. *Assuming UGC to be true, the* Above Guarantee Vertex Cover *problem in graphs with a perfect matching cannot be approximated to within a factor of* c*, for any constant* $c > 1$.

Proof. Suppose that there exists a c-approximate algorithm $\mathcal{A}$ for Above Guarantee Vertex Cover on graphs with a perfect matching for some constant $c > 1$. By Lemmas 4.7 and 4.8, it is sufficient to exhibit a $(2-\epsilon)$-approximate algorithm, for some constant $\epsilon > 0$, for VC-PM. This would give us the desired contradiction. We show that $\mathcal{A}$ itself is such an algorithm and obtains a $(2-\epsilon)$-approximate solution with $\epsilon = 2/(c+1)$.

Let $G = (V, E)$ be a graph on $2n$ vertices with a perfect matching and minimum vertex cover of size $n + \alpha n$, where $1/n \leq \alpha \leq 1$. Use algorithm $\mathcal{A}$ on G to obtain a vertex cover of size at most $n + c\alpha n$. The quality of this solution is $(n+c\alpha n)/(n+\alpha n) = (1+c\alpha)/(1+\alpha)$. We distinguish two cases: (1) $\alpha < 1/c$ and (2) $\alpha \geq 1/c$. We claim that in either case the approximation factor is $2 - 2/(c+1) = 2c/(c+1)$. To see this, first consider the case when $\alpha < 1/c$. It is straightforward to show that $(1+c\alpha)/(1+\alpha) < 2c/(c+1)$ if and only if $\alpha < 1/c$. When $\alpha \geq 1/c$, note that $\mathcal{A}$ returns the entire vertex set of G as solution. The approximation factor in this case is $2/(1+\alpha)$ which can be easily seen to be at most $2c/(c+1)$. This completes the proof of the theorem. □

Since there is an approximation-preserving reduction (Lemma 4.5) from Above Guarantee Vertex Cover to Min 2-Cnf Sat Del, a constant-factor approximation algorithm for the latter implies the existence of a constant-factor approximation algorithm for the former. Thus we have,

Corollary 4.1. *If UGC is true then* Min 2-Cnf Sat Del *does not admit a* c*-factor approximation algorithm, for any constant* $c > 1$.

To the best of our knowledge, the only other hardness result known for Min 2-Cnf Sat Del is a 2.88-approximation hardness assuming P $\neq$ NP due to Chlebik and Chlebikova [35].

Dinur and Safra [45] have shown that unless P = NP, Vertex Cover cannot be approximated to within 1.3606 even on graphs with a perfect matching. Using this, we obtain:

Corollary 4.2. *Under the hypothesis* P ≠ NP*,* Above Guarantee Vertex Cover *in graphs with a perfect matching cannot be approximated to within* 1.7212*.*

Proof. Let $\mathcal{A}$ be a d-approximation algorithm for computing $\tau(G)$ in graphs with a perfect matching. Let G be an n-vertex graph with a perfect matching. Using $\mathcal{A}$, one can obtain a vertex cover of size at most $n/2 + d\tau(G)$. An optimum vertex cover of G has size $n/2 + \tau(G)$. By the NP-hardness of approximating Vertex Cover [45], we must have $(n + 2d\tau(G))/(n + 2\tau(G)) \geq 1.3606$. Simplifying this yields $n/\tau(G) \leq 2(d - 1.3606)/0.3606$. Note that $n/\tau(G) \geq 2$ and so $d \geq 1.7212$. □

4.4 The Kőnig Vertex Deletion Problem

Recall that the Kőnig Vertex Deletion problem is, given a graph and a nonnegative integer parameter k, to decide whether there exist at most k vertices whose deletion makes the resulting graph Kőnig. We first investigate the parameterized complexity of this problem and then describe an approximation algorithm for its optimization version.

4.4.1 Parameterized Complexity

We first consider the case when the input graph has a perfect matching.

Graphs with a Perfect Matching

For graphs with a perfect matching it turns out that Kőnig Vertex Deletion and AGVC are fixed-parameter equivalent.

Theorem 4.7. *Let G be an n-vertex graph with a perfect matching. Then G has a vertex cover of size at most $n/2+k$ if and only if G has a Kőnig vertex deletion set of size at most $2k$.*

Proof. ($\Rightarrow$) Let P be a perfect matching of G and C a vertex cover of G of size at most $n/2 + k$. Consider the subset $M \subseteq P$ of matching edges both of whose endpoints are in C. Clearly $V[M]$ is a Kőnig vertex deletion set of G of size at most $2k$.

($\Leftarrow$) Conversely let K be a Kőnig vertex deletion set of G of size $r \leq 2k$. Then $G \setminus K$ is a Kőnig graph on $n - r$ vertices and hence has a vertex cover C' of size at most $(n - r)/2$. Clearly $C = C' \cup K$ is a vertex cover of G of size $|C'| + |K| \leq (n - r)/2 + r = (n + r)/2 \leq n/2 + k$. □

The following corollary follows from Theorems 4.1 and 4.7 and the fact that Vertex Cover is NP-complete.

Corollary 4.3. *The* Kőnig Vertex Deletion *problem is* NP*-complete.*

By Theorem 4.3, AGVC is fixed-parameter tractable and therefore we have:

Corollary 4.4. *The* Kőnig Vertex Deletion *problem, parameterized by the solution size, is fixed-parameter tractable on graphs with a perfect matching.*

The next result relates the size of a minimum vertex cover with that of a minimum Kőnig vertex deletion set for graphs with a perfect matching.

Corollary 4.5. *Let G be an n-vertex graph with a perfect matching P. Then $\beta(G) = n/2 + k$ if and only if $\kappa(G) = 2k$. Moreover if $\kappa(G) = 2k$, then there exists an edge subset $M \subseteq P$ of size k such that $V[M]$ is a minimum Kőnig vertex deletion set of G.*

If we let $\tau(G) = \beta(G) - \mu(G)$, then the above corollary states: $\kappa(G) = 2\tau(G)$.

Graphs Without a Perfect Matching

For graphs without a perfect matching we do not know of a reduction from Kőnig Vertex Deletion to AGVC and neither does the general case seem reducible to the case where the graph has a perfect matching. However we show that the general problem is fixed-parameter tractable using some new structural results between maximum matchings and vertex covers.

To begin with, we derive a weaker version of Theorem 4.7 which relates the size of a vertex cover with that of a Kőnig vertex deletion set for graphs without a perfect matching.

Theorem 4.8. *Let G be a graph without a perfect matching. If G has a vertex cover of size $\mu(G) + k$ then G has a Kőnig vertex deletion set of size at most $2k$. Moreover, $\tau(G) \leq \kappa(G) \leq 2\tau(G)$, where $\tau(G) = \beta(G) - \mu(G)$.*

Proof. Let M be a maximum matching of G and let C be a vertex cover of G of size $\mu(G) + k$. Define $I = V \setminus V[M]$, $C_I = C \cap I$ and M' to be the subset of M both of whose endpoints are in C. Clearly $V[M'] \cup C_I$ is a Kőnig vertex deletion set of G of size at most $2k$. This shows that $\kappa(G) \leq 2\tau(G)$. To prove that $\tau(G) \leq \kappa(G)$, suppose that there exists $S \subseteq V$, $|S| < \tau(G)$, such that $G \setminus S$ is Kőnig. Then the following easily verifiable inequalities:

$$\begin{aligned} \mu(G \setminus S) &\leq \mu(G) \\ \beta(G \setminus S) &\geq \beta(G) - |S| = \mu(G) + \tau(G) - |S| \end{aligned}$$

imply that $\beta(G \setminus S) > \mu(G \setminus S)$, a contradiction. □

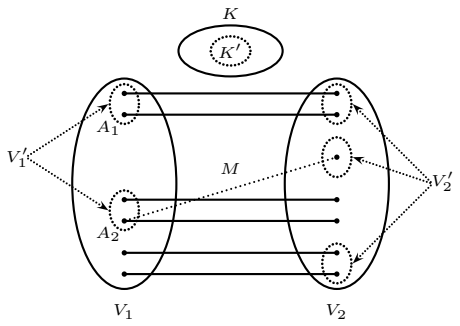


Figure 4.2: The sets that appear in the proof of Theorem 4.9. The matching M consists of the solid edges across V_1 and V_2.

Suppose Y is a vertex cover in a graph $G = (V, E)$. Consider a maximum matching M between Y and $V \setminus Y$. If M saturates every vertex of Y then the graph is Kőnig. If not, then $Y \setminus V(M)$, the set of vertices of Y unsaturated by M, is a Kőnig vertex deletion set by Lemma 4.2. What we prove in this section is that if Y is a minimum vertex cover, then $Y \setminus V(M)$ is a minimum Kőnig vertex deletion set. Our first observation is that any minimum Kőnig vertex deletion set is contained in some minimum vertex cover.

Theorem 4.9. *Let G be a graph with a minimum Kőnig vertex deletion set K. Let $V(G \setminus K) = V_1 \uplus V_2$ where V_2 is independent and there is a matching M from V_1 to V_2 saturating V_1. Then $V_1 \cup K$ is a minimum vertex cover for G.*

Proof. Suppose S is a vertex cover of G such that $|S| < |V_1| + |K|$. We will show that there exists a Kőnig vertex deletion set of size smaller than $|K|$, contradicting our hypothesis. Define $V_1' = V_1 \cap S$, $V_2' = V_2 \cap S$ and $K' = K \cap S$. Let A_1 be the vertices of V_1' whose partner in M is in V_2' and let A_2 be the vertices of V_1' whose partner in M is not in V_2'. See Figure 4.2. We claim that $A_1 \cup K'$ is a Kőnig vertex deletion set of G and $|A_1 \cup K'| < |K|$, which will produce the required contradiction and prove the theorem. This claim is proved using the following three claims:

Claim 1. $|A_1 \cup K'| < |K|$.

Claim 2. *$A_2 \cup V_2'$ is a vertex cover in $G \setminus (A_1 \cup K')$.*

Claim 3. *There exists a matching between $A_2 \cup V_2'$ and $V \setminus (V_1' \cup K' \cup V_2')$ saturating every vertex of $A_2 \cup V_2'$.*

Proof of Claim 1. Clearly $|S| = |V_1'| + |V_2'| + |K'|$. Note that S intersects $|A_1|$ of the edges of M in both end points and $|M| - |A_1|$ edges of M in one end point (in either V_1' or V_2'). Furthermore V_2' has $|V_2' \setminus V(M)|$ vertices of S

that do not intersect any edge of M. Hence

$$|M| + |A_1| + |V_2' \setminus V(M)| = |V_1'| + |V_2'|.$$

That is,

$$|V_1| + |A_1| + |V_2' \setminus V(M)| = |V_1'| + |V_2'|,$$

as $|M| = |V_1|$. Hence $|S| < |V_1| + |K|$ implies that

$$|A_1| + |V_2' \setminus V(M)| + |K'| < |K|$$

which implies that $|A_1| + |K'| < |K|$ proving the claim.

Proof of Claim 2. Since $S = A_1 \cup A_2 \cup V_2' \cup K'$ is a vertex cover of G, clearly $A_2 \cup V_2'$ covers all edges in $G \setminus (A_1 \cup K')$.

Proof of Claim 3. Since the partner of a vertex in A_2 in M is in $V \setminus (V_1' \cup K' \cup V_2')$, we can use the edges of M to saturate vertices in A_2. To complete the proof, we show that in the bipartite graph

$$B = (V_2', (V_1 \setminus V_1') \cup (K \setminus K'))$$

there is a matching saturating V_2'. To see this, note that any subset $D \subseteq V_2'$ has at least $|D|$ neighbors in $(V_1 \setminus V_1') \cup (K \setminus K')$. For otherwise, let D' be the set of neighbors of D in $(V_1 \setminus V_1') \cup (K \setminus K')$ where we assume $|D| > |D'|$. Then $(S \setminus D) \cup D'$ is a vertex cover of G of size strictly less than $|S|$, contradicting the fact that S is a minimum vertex cover. To see that $(S \setminus D) \cup D'$ is indeed a vertex cover of G, note that $S \setminus V_2'$ covers all edges of G except those in the graph B and all these edges are covered by $(V_2' \setminus D) \cup D'$. Hence by Hall's theorem, there exists a matching saturating all vertices of V_2' in the bipartite graph B, proving the claim.

This completes the proof of the theorem. □

Theorem 4.9 has interesting consequences.

Corollary 4.6. *If K_1 and K_2 are minimum Kőnig vertex deletion sets of G, then $\mu(G \setminus K_1) = \mu(G \setminus K_2)$.*

Proof. Since K_1 and K_2 are minimum Kőnig vertex deletion sets of G, $\beta(G \setminus K_1) = \mu(G \setminus K_1)$ and $\beta(G \setminus K_2) = \mu(G \setminus K_2)$. By Theorem 4.9, $\beta(G \setminus K_1) + |K_1| = \beta(G)$ and $\beta(G \setminus K_2) + |K_2| = \beta(G)$. Since $|K_1| = |K_2|$, it follows that $\beta(G \setminus K_1) = \beta(G \setminus K_2)$ and hence $\mu(G \setminus K_1) = \mu(G \setminus K_2)$. □

From Theorem 4.9 and Lemma 4.4, we get

Corollary 4.7. *Given a graph $G = (V, E)$ and a minimum Kőnig vertex deletion set for G, one can construct a minimum vertex cover for G in polynomial time.*

Our goal now is to prove the "converse" of Corollary 4.7. In particular, we would like to construct a minimum Kőnig vertex deletion set from a minimum vertex cover. Our first step is to show that if we know that a given minimum vertex cover contains a minimum Kőnig vertex deletion set then we can find the Kőnig vertex deletion set in polynomial time. Recall that given a graph $G = (V, E)$ and $A, B \subseteq V$ such that $A \cap B = \emptyset$, we use $\mu(A, B)$ to denote a maximum matching in the bipartite graph comprising of the vertices in $A \cup B$ and the edges in $\{\{u, v\} \in E : u \in A, v \in B\}$. We denote this graph by (A, B).

Lemma 4.9. *Let K be a minimum Kőnig vertex deletion set and Y a minimum vertex cover of a graph $G = (V, E)$ such that $K \subseteq Y$. Then $\mu(G \setminus K) = \mu(Y, V \setminus Y)$ and $|K| = |Y| - \mu(Y, V \setminus Y)$.*

Proof. If G is Kőnig then the theorem clearly holds. Therefore assume that $K \neq \emptyset$. Note that $Y \setminus K$ is a minimum vertex cover of the Kőnig graph $G \setminus K$. Thus $\mu(G \setminus K) = \mu(Y \setminus K, V \setminus Y)$. We claim that $\mu(Y \setminus K, V \setminus Y) = \mu(Y, V \setminus Y)$. For if not, we must have $\mu(Y \setminus K, V \setminus Y) < \mu(Y, V \setminus Y)$. Then let M be a maximum matching in the bipartite graph $(Y, V \setminus Y)$ and $K' \subseteq Y$ be the set of vertices unsaturated by M. Note that $K' \neq \emptyset$ is a Kőnig vertex deletion set for G. Since $\mu(Y, V \setminus Y) = |Y| - |K'|$ and $\mu(Y \setminus K, V \setminus Y) = |Y| - |K|$ we have $|K'| < |K|$, a contradiction, since by hypothesis K is a smallest Kőnig vertex deletion set for G. Therefore we must have $\mu(G \setminus K) = \mu(Y, V \setminus Y)$ and $|K| = |Y| - \mu(Y, V \setminus Y)$. □

The next lemma says that $\mu(Y, V \setminus Y)$ is the same for all minimum vertex covers Y of a graph G. Together with Lemma 4.9, this implies that if K is a minimum Kőnig vertex deletion set and Y is a minimum vertex cover of a graph $G = (V, E)$, then $\mu(G \setminus K) = \mu(Y, V \setminus Y)$.

Lemma 4.10. *For any two minimum vertex covers Y_1 and Y_2 of G, $\mu(Y_1, V \setminus Y_1) = \mu(Y_2, V \setminus Y_2)$.*

Proof. Suppose without loss of generality that $\mu(Y_1, V \setminus Y_1) > \mu(Y_2, V \setminus Y_2)$. Let M_1 be a maximum matching in the bipartite graph $(Y_1, V \setminus Y_1)$. To arrive at a contradiction, we study how Y_2 intersects the sets Y_1 and $V \setminus Y_1$ with respect to the matching M_1. To this end, we define the following sets (see Figure 4.3):

- $A = Y_2 \cap Y_1 \cap V(M_1)$.
- $B = Y_2 \cap (V \setminus Y_1) \cap V(M_1)$.
- A_1 is the set of vertices in A whose partners in M_1 are also in Y_2.
- A_2 is the set of vertices in A whose partners in M_1 are not in Y_2.

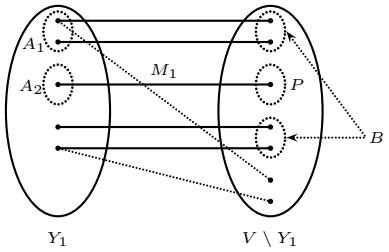


Figure 4.3: The sets that appear in the proof of Lemma 4.10. The solid edges across Y_1 and $V \setminus Y_1$ constitute the matching M_1.

We first show that

Claim. In the bipartite graph $(Y_2, V \setminus Y_2)$ there is a matching saturating each vertex in $A_2 \cup B$.

It will follow from the claim that $\mu(Y_2, V \setminus Y_2) \geq |A_2| + |B|$. However, note that Y_2 intersects every edge of M_1 at least once (as Y_2 is a vertex cover). More specifically, Y_2 intersects $|A_1|$ edges of M_1 twice and $|M_1| - |A_1|$ edges once (either in Y_1 or in $V \setminus Y_1$). Hence, $|A| + |B| = |M_1| + |A_1|$ and so $|A_2| + |B| = |M_1|$ and so $\mu(Y_2, V \setminus Y_2) \geq |A_2| + |B| = |M_1|$ a contradiction to our assumption at the beginning of the proof. Thus it suffices to prove the claim.

Proof of Claim. Let P denote the partners of the vertices of A_2 in M_1. Since $P \subseteq V \setminus Y_2$, we use the edges of M_1 to saturate vertices of A_2. Hence it is enough to show that the bipartite graph $\mathcal{B} = (B, (V \setminus Y_2) \setminus P)$ contains a matching saturating the vertices in B. Suppose not. By Hall's Theorem there exists a set $D \subseteq B$ such that $|N_{\mathcal{B}}(D)| < |D|$. We claim that the set $Y_2' := Y_2 \setminus D + N_{\mathcal{B}}(D)$ is a vertex cover of G. To see this, note that the vertices in $Y_2 \setminus D$ cover all the edges of G except those in the bipartite graph $(D, Y_1 \cap (V \setminus Y_2))$ and these are covered by $N_{\mathcal{B}}(D)$. Therefore Y_2' is a vertex cover of size strictly smaller than Y_2, a contradiction. This proves that there exists a matching in $(Y_2, V \setminus Y_2)$ saturating each vertex in $A_2 \cup B$.

This completes the proof of the lemma. □

The next theorem shows how we can obtain a minimum Kőnig vertex deletion set from a minimum vertex cover in polynomial time.

Theorem 4.10. *Given a graph $G = (V, E)$, let Y be any minimum vertex cover of G and M a maximum matching in the bipartite graph $(Y, V \setminus Y)$. Then $K := Y \setminus V(M)$ is a minimum Kőnig vertex deletion set of G.*

Proof. Clearly K is a Kőnig vertex deletion set. Let K_1 be a minimum Kőnig vertex deletion set of G. By Theorem 4.9, there exists a minimum

vertex cover Y_1 such that $K_1 \subseteq Y_1$ and

$$\begin{array}{rcl} |K_1| &=& |Y_1| - \mu(Y_1, V \setminus Y_1) \quad \text{(By Lemma 4.9.)} \\ &=& |Y| - \mu(Y_1, V \setminus Y_1) \\ &=& |Y| - \mu(Y, V \setminus Y) \quad \text{(By Lemma 4.10.)} \\ &=& |K| \end{array}$$

The second equality holds since Y_1 and Y are minimum vertex covers. This proves that K is a minimum Kőnig vertex deletion set. □

Corollary 4.8. *Given a graph $G = (V, E)$ and a minimum vertex cover for G, one can construct a minimum Kőnig vertex deletion set for G in polynomial time.*

Note that although both these problems—Vertex Cover and Kőnig Vertex Deletion—are NP-complete, we know of very few pairs of such parameters where we can obtain one from the other in polynomial time on the same graph (e.g. edge dominating set and minimum maximal matching, see [66]). In fact, there are parameter pairs such as dominating set and vertex cover where such a polynomial-time transformation is not possible unless P = NP. This follows since in bipartite graphs, for instance, a minimum vertex cover is computable in polynomial time whereas computing a minimum dominating set is NP-complete.

We are now ready to prove that the Kőnig Vertex Deletion problem is fixed-parameter tractable in general graphs.

Theorem 4.11. *Given a graph $G = (V, E)$ and an integer parameter k, the problem of whether G has a subset of at most k vertices whose deletion makes the resulting graph Kőnig can be decided in time $O(15^k \cdot k^2 \cdot |E|^3)$. Hence the* Kőnig Vertex Deletion *problem is fixed-parameter tractable when parameterized by the solution size.*

Proof. Use the FPT-algorithm from Theorem 4.3 to test whether G has a vertex cover of size at most $\mu(G) + k$. If not, by Theorem 4.8, we know that the size of a minimum Kőnig vertex deletion set is strictly more than k. Therefore return no. If yes, then find the size of a minimum vertex cover by applying Theorem 4.3 with every integer value between 0 and k for the excess above $\mu(G)$. Note that for yes-instances of the Above Guarantee Vertex Cover problem, the FPT-algorithm actually outputs a vertex cover of size $\mu(G) + k$. We therefore obtain a minimum vertex cover of G. Use Theorem 4.10 to get a minimum Kőnig vertex deletion set in polynomial time and depending on its size answer the question. It is easy to see that all this can be done in time $O(15^k \cdot k^2 \cdot |E|^3)$. □

Note that computing a maximum independent set (or equivalently a minimum vertex cover) in an n-vertex graph can be done in time $O^*(2^{0.288n})$ [62].

By Corollary 4.8, one can compute a minimum Kőnig vertex deletion set in the same exponential time.

Corollary 4.9. *Given a graph $G = (V, E)$ on n vertices one can find a minimum Kőnig vertex deletion set in time $O^*(2^{0.288n}) = O^*(1.221^n)$.*

Suppose we wanted to compute a minimum Kőnig vertex deletion set on graphs of treewidth at most w. A dynamic programming approach as for Dominating Set or Independent Set is not obvious. However since one can obtain a minimum vertex cover on graphs with treewidth at most w in time $O^*(2^w)$ [107], by Corollary 4.8, one can obtain a minimum Kőnig deletion set within this time.

Corollary 4.10. *If a tree-decomposition for G of width w is given, one can find a minimum Kőnig vertex deletion set in time $O^*(2^w)$.*

4.4.2 Approximability

In Theorem 4.8 we established that for any graph G (whether it has a perfect matching or not), $\tau(G) \leq \kappa(G) \leq 2\tau(G)$, where $\tau(G) = \beta(G) - \mu(G)$ is the excess vertex cover beyond the size of a maximum matching. Therefore a good approximation of the above guarantee parameter τ yields a good approximation for κ and vice versa.

In the algorithm outlined in Figure 4.1, note that $V(\mathcal{S}) \cap V(G)$ is actually a Kőnig vertex deletion set of G. Since $|V(\mathcal{S})| \leq O(\log n \log\log n)\ (\beta - \mu)$, we have, by Theorem 4.4

Theorem 4.12. *Given a graph G on n vertices, there exists an algorithm that approximates the Kőnig vertex deletion set of G to within a factor of $O(\log n \log\log n)$.*

On graphs with a perfect matching, the Above Guarantee Vertex Cover and Kőnig Vertex Deletion problems are equivalent and hence Theorems 4.6 and 4.7 imply

Corollary 4.11. *If UGC is true then* Kőnig Vertex Deletion *does not admit a constant-factor approximation algorithm.*

Since Kőnig Vertex Deletion and Above Guarantee Vertex Cover are equivalent in terms of approximability in graphs with a perfect matching (Theorem 4.7), Corollary 4.2 implies

Corollary 4.12. *Under the hypothesis* P $\neq$ NP, Kőnig Vertex Deletion *in graphs with a perfect matching cannot be approximated to within* 1.7212.

4.5 The Induced Kőnig Subgraph Problem

In this section we deal with the parameterized complexity and approximability of the vertex and edge versions of the Induced Kőnig Subgraph problem.

4.5.1 Vertex Induced Kőnig Subgraph

The NP-completeness of this problem follows from that of Kőnig Vertex Deletion but it has a different parameterized complexity. We show that Vertex Induced Kőnig Subgraph is W[1]-hard and is as hard to approximate as the Independent Set problem.

Theorem 4.13. Vertex Induced Kőnig Subgraph *is* W[1]*-hard with respect to the number of vertices in the induced subgraph as parameter.*

Proof. We give a parameter-preserving reduction from Independent Set to Vertex Induced Kőnig Subgraph. Given an instance (G, k) of Independent Set, construct a graph H as follows. The vertex set of H consists of two copies of $V(G)$ namely, $V_1 = \{u_1 \colon u \in V(G)\}$ and $V_2 = \{u_2 \colon u \in V(G)\}$. For all $u \in V(G)$, $(u_1, u_2) \in E(H)$. If $(u, v) \in E(G)$, add the edges (u_1, v_1), (u_2, v_2), (u_1, v_2) and (v_1, u_2) in $E(H)$. H has no more edges.

We claim that G has an independent set of size k if and only if H has a Kőnig subgraph of size $2k$. Let I be an independent set of size k in G. Let $K = \{u_1, u_2 \in V(H) \colon u \in I\}$. Clearly $H[K]$ is an induced matching on $2k$ vertices and is bipartite and hence Kőnig. Conversely, let K be a Kőnig subgraph of H on $2k$ vertices. By Lemma 4.2, every Kőnig graph on n vertices has an independent set of size at least $n/2$. Therefore let I' be an independent set of K of size at least k. Define $I = \{u \in V(G) \colon \text{either } u_1 \text{ or } u_2 \in I'\}$. It is clear that the vertices of I' correspond to distinct vertices of G and hence $|I| \geq k$. It is also easy to see that the vertices in I actually form an independent set in G. □

Since the Independent Set problem can have no approximation algorithms with factor $O(n^{1-\epsilon})$, for any $\epsilon > 0$, unless P = NP [80, 127], we have:

Corollary 4.13. *The* Vertex Induced Kőnig Subgraph *problem cannot be approximated in polynomial time to within a factor of* $O(n^{1-\epsilon})$*, for any* $\epsilon > 0$*, unless* P = NP*.*

In the reduction above, $|V(H)| = 2|V(G)|$ and (G, k) is a yes-instance of Independent Set if and only if $(H, 2k)$ is a yes-instance of Vertex Induced Kőnig Subgraph. Thus this reduction can also be viewed as a reduction from Vertex Cover to Kőnig Vertex Deletion giving yet another proof of Corollary 4.3.

4.5.2 Edge Induced Kőnig Subgraph

In this section we study the Edge Induced Kőnig Subgraph problem. We begin by showing that it is NP-complete.

NP-Completeness

Since both Kőnig Edge Deletion and Edge Induced Kőnig Subgraph have the same complexity from the classical point of view, it is sufficient to prove that one of them is NP-complete. We actually show that:

Theorem 4.14. Kőnig Edge Deletion *is* NP*-complete.*

Proof. We give a reduction from Min 2-Cnf Sat Del. Let Φ be a 2-Cnf Sat formula with m clauses composed of the literals $\{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$. Construct a graph $G_\Phi = (V, E)$ as follows. The vertex set V consists of $m+3$ copies of $x_i, \bar{x}_i$, for $1 \leq i \leq n$:

$$x_i, \bar{x}_i, x_{i1}, \bar{x}_{i1}, \ldots, x_{i,m+2}, \bar{x}_{i,m+2}.$$

Add exactly those edges so that for each $1 \leq i \leq n$, the vertex sets $L_i = \{x_i, x_{i,1}, \ldots, x_{i,m+2}\}$ and $R_i = \{\bar{x}_i, \bar{x}_{i,1}, \ldots, \bar{x}_{i,m+2}\}$ form a complete bipartite graph with L_i and R_i as the left and right partite sets, respectively. Finally for each clause $(y_i \vee y_j)$ of Φ add an edge (y_i, y_j) (among the vertices $\{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$). Note that G_Φ has a perfect matching and that each clause of Φ corresponds to an edge of G_Φ.

Claim 4.1. *There exists an assignment satisfying all but k clauses of Φ if and only if there exist at most k edges whose deletion makes G_Φ Kőnig.*

($\Rightarrow$) Let α be an assignment to the variables of Φ that satisfies all but k clauses. Each of these k clauses corresponds to a distinct edge in G_Φ. Delete these edges from G_Φ. Then for each edge in the remaining graph, at least one endpoint of the edge is assigned 1 by the assignment α. To prove that the remaining graph is Kőnig, by Lemma 4.2, we must demonstrate a bipartition of the vertex set into $V_1 \uplus V_2$ (say) such that V_2 is independent and there exists a matching across the cut (V_1, V_2) saturating V_1. If $\alpha(x_i) = 1$ then place the vertices $x_i, x_{i,1}, \ldots x_{i,m+2}$ in V_1; else place $\bar{x}_i, \bar{x}_{i,1}, \ldots \bar{x}_{i,m+2}$ in V_1. The remaining vertices are placed in V_2. As Φ satisfies all remaining clauses, V_2 is independent. Note that if $x_i \in V_1$ then $\bar{x}_i \in V_2$ and vice versa. Also if $x_{i,j} \in V_1$ then $\bar{x}_{i,j} \in V_2$ and vice versa. Hence there exists a matching across the cut (V_1, V_2) that saturates V_1.

($\Leftarrow$) Conversely suppose that deleting a set S of k edges makes G_Φ Kőnig. We will assume that S is a *minimal* edge deletion set. Any minimal Kőnig edge deletion set has size at most m, since deleting all the m "clause edges" from G_Φ results in a Kőnig graph. Therefore we may assume that $k \leq m$. Call the resulting graph G'_Φ. Then the vertex set of G'_Φ can be partitioned

into V_1 and V_2 such that V_2 is independent and there exists a matching across the cut (V_1, V_2) that saturates V_1.

Claim 1. For each $1 \leq i \leq n$, it is not the case that $x_i, \bar{x}_i \in V_2$.

Suppose that for some $1 \leq i \leq n$, $x_i, \bar{x}_i \in V_2$. Then it must be that $\bar{C}_i = \{\bar{x}_{i,1}, \ldots, \bar{x}_{i,m+2}\} \not\subseteq V_2$ since otherwise $m+2$ edges between x_i and the vertices of $\bar{C}_i$ must have been deleted from G_Φ to obtain G'_Φ, a contradiction. If $\bar{C}_i \subseteq V_1$ then $C_i = \{x_{i,1}, \ldots, x_{i,m+2}\} \subseteq V_2$ for there to be a matching across the cut (V_1, V_2) saturating all of $\bar{C}_i$. But then $m+2$ edges between $\bar{x}_i$ and C_i must have been deleted from G_Φ to obtain G'_Φ, again a contradiction. This argument shows that that there exist integers $p, q \geq 1$ with $p + q = m+2$, such that V_1 contains p vertices of $\bar{C}_i$ and V_2 contains the remaining q vertices of $\bar{C}_i$. In order for there to be a matching across (V_1, V_2) saturating all p vertices of $\bar{C}_i$ in V_1 there must be at least p vertices of C_i in V_2. Since the vertices of C_i and $\bar{C}_i$ form a complete bipartite graph we end up deleting at least $pq + 1 \geq m + 2$ edges of G_Φ, a contradiction yet again. This proves Claim 1.

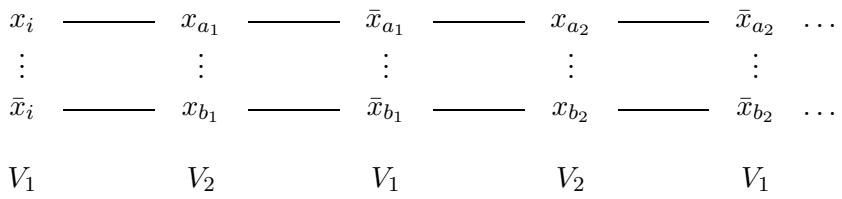
Claim 2. For each $1 \leq i \leq n$, it is not the case that $x_i, \bar{x}_i \in V_1$.

Suppose that there exists i, $1 \leq i \leq n$, such that $x_i, \bar{x}_i \in V_1$. Let M be a matching across (V_1, V_2) that saturates the vertices of V_1. We distinguish three cases.

Case 1. $\bar{C}_i = \{\bar{x}_{i,1}, \ldots, \bar{x}_{i,m+2}\} \subseteq V_1$.

This implies that $C_i = \{x_{i,1}, \ldots, x_{i,m+2}\} \subseteq V_2$ as otherwise no matching across (V_1, V_2) would saturate all of $\bar{C}_i$. Let x_{a_1} and x_{b_1} be the partners of x_i and $\bar{x}_i$, respectively, relative to the matching M. By Claim 1, x_{a_1} and x_{b_1} represent different variables, that is, they are not the negations of one another. This implies that $\bar{x}_{a_1}$ and $\bar{x}_{b_1}$ are in V_1. Consider the pair $x_{a_1}, \bar{x}_{a_1}$. We will show that $C_{a_1} = \{x_{a_1,1}, \ldots, x_{a_1,m+2}\} \subseteq V_2$ and $\bar{C}_{a_1} = \{\bar{x}_{a_1,1}, \ldots, \bar{x}_{a_1,m+2}\} \subseteq V_1$. For if not, suppose that $1 \leq q \leq m+1$ vertices of $\bar{C}_{a_1}$ are in V_2 while the remaining $p \geq 1$ vertices of $\bar{C}_{a_1}$ are in V_1. In order for the vertices of $\bar{C}_{a_1}$ to have partners with respect to M at least p vertices of C_{a_1} must be in V_2. This implies that at least pq edges have been deleted from G_Φ to obtain G'_Φ. Since $p + q = m + 2$, we have $pq \geq m + 1$, a contradiction. The upshot is that the partners of $\bar{x}_{a_1}$ and $\bar{x}_{b_1}$ relative to M are vertices from the set $\{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$. Let the partners of $\bar{x}_{a_1}$ and $\bar{x}_{b_1}$ relative to M be x_{a_2} and x_{b_2} respectively. Again by Claim 1, x_{a_2} and x_{b_2} represent distinct variables and hence $\bar{x}_{a_2}$ and $\bar{x}_{b_2}$ are in V_1. Repeating this

argument we obtain a sequence of vertices of the form:



Since there are only $2n$ vertices such a chain must end at V_2 with both endpoints being the negation of one another. This contradicts Claim 1 and shows that this situation does not arise.

Case 2. $\bar{C}_i = \{\bar{x}_{i,1}, \ldots, \bar{x}_{i,m+2}\} \subseteq V_2$ and $C_i = \{x_{i,1}, \ldots, x_{i,m+2}\} \subseteq V_1$.
This is the symmetric version of Case 1 and can be handled similarly.

Case 3. For some integers $p, q \geq 1$ and $p + q = m + 2$, p vertices of $\bar{C}_i$ lie in V_1 and the remaining q vertices lie in V_2.

By symmetry, p vertices of C_i must lie in V_2. This implies that at least $pq \geq m + 1$ edges have been deleted from G_Φ to obtain G'_Φ, a contradiction. This proves Claim 2.

Since S was assumed to be a minimal Kőnig edge deletion set, for each vertex y_i, all copies $y_{i,1}, \ldots, y_{i,m+2}$ of it must be placed in the same partition as y_i itself and hence all edges of G_Φ that are part of the n copies of $K_{m+3,m+3}$ lie across the cut (V_1, V_2). It is easy to see that any other partitioning of the copies would result in more edges being deleted unnecessarily. Therefore the edges that were deleted from G_Φ were those that corresponded to the clauses of Φ. If a vertex y_i is in V_1 assign the corresponding literal the value 1; else assign the literal the value 0. Note that this assignment is consistent as all copies of a vertex are in the same partition as the vertex itself and for no vertex do we have that $x_i, \bar{x}_i \in V_1$ or $x_i, \bar{x}_i \in V_2$. This assignment satisfies all but the k clauses that correspond to the edges that were deleted. □

Since the above reduction is cost-preserving, an approximation lower-bound for Min 2-Cnf Sat Del is a lower-bound for Kőnig Edge Deletion too. Therefore by Corollary 4.1, we obtain

Corollary 4.14. *If UGC is true then* Kőnig Edge Deletion *does not have a c-approximation algorithm, for any constant* $c > 1$.

Chlebik and Chlebikova [35] have shown that it is NP-hard to approximate Min 2-Cnf Sat Del to within $8\sqrt{5} - 15 \approx 2.88$. This gives us

Corollary 4.15. *It is* NP*-hard to approximate* Kőnig Edge Deletion *to within* 2.88.

Approximation Results

For Edge Induced Kőnig Subgraph, an easy 2-approximation algorithm is as follows: Find a cut of size $m/2$ and delete all other edges; the resulting graph is bipartite and hence Kőnig. In this subsection, we give a 4/3-approximation algorithm for graphs with a perfect matching and a 5/3-approximation algorithm for general graphs based on the following combinatorial results.

Theorem 4.15. *Let $G = (V, E)$ be a graph with a maximum matching M and let $G_M = (V_M, E_M)$ be the graph induced on the vertices $V(M)$ of M. Then G has an edge-induced Kőnig subgraph of size at least*

$$\frac{3|E_M|}{4} + \frac{|E - E_M|}{2} + \frac{|M|}{4}.$$

Proof. Randomly partition the vertex set of G into $V_1 \uplus V_2$ as follows. For each edge $e_i \in M$, select an endpoint of e_i with probability 1/2 and place it in V_1. Define $V_2 = V - V_1$. Note that the edges in M always lie across the cut (V_1, V_2). An edge of $E_M - M$ is in $G[V_2]$ with probability 1/4; an edge in $E - E_M$ lies in $G[V_2]$ with probability 1/2. For each edge $e \in E$, define X_e to be the indicator random variable that takes the value 1 if $e \in G[V_2]$ and 0 otherwise. Also define $X = \sum_{e \in E} X_e$. Then

$$E[X] = \sum_{e \in E} E[X_e] = \frac{|E_M - M|}{4} + \frac{|E - E_M|}{2}.$$

Deleting the edges in $G[V_2]$ results in a Kőnig graph with

$$\frac{3|E_M|}{4} + \frac{|E - E_M|}{2} + \frac{|M|}{4}$$

edges in expectation. This algorithm can be easily derandomized by the method of conditional probabilities (see, for instance [104]). This completes the proof. □

If $G = (V, E)$ has a perfect matching M then $E_M = E$ and $|M| = |V|/2$ and we have

Corollary 4.16. *Let $G = (V, E)$ be a graph on n vertices and m edges with a perfect matching. Then G has a subgraph with at least $3m/4 + n/8$ edges that is Kőnig. This subgraph can be found in time $O(mn)$.*

Theorem 4.16. *Let $G = (V, E)$ be an undirected graph on n vertices and m edges. Then G has an edge-induced Kőnig subgraph of size at least $3m/5$.*

Proof. Let M be a maximum matching of G and let $G[V_M] = (V_M, E_M)$ be the subgraph induced by the vertices $V(M)$ of M. Let $\eta(G)$ denote the size of the maximum edge induced Kőnig subgraph of G. By Theorem 4.15,

$$\eta(G) \geq \frac{|E_M| + |M|}{4} + \frac{|E|}{2}.$$

Observe that by deleting all the edges in $G[V_M]$ we obtain a Kőnig subgraph of G. In fact, this is a bipartite graph with bipartition V_M and $V - V_M$. Therefore if $|E - E_M| \geq 3m/5$, the statement of the theorem clearly holds. Otherwise, $|E_M| \geq 2m/5$ and by Theorem 4.15, we obtain $\eta(G) \geq |M|/4 + 3m/5$. This completes the proof. □

The following theorem follows from Corollary 4.16 and Theorem 4.16 and the fact that the optimum Kőnig subgraph has at most m edges.

Theorem 4.17. *The optimization version of* Edge Induced Kőnig Subgraph *is approximable to within a factor of* 5/3 *for general graphs. This factor can be improved to* 4/3 *when restricted to graphs with a perfect matching.*

FPT Algorithms

Note that Theorem 4.16 actually shows that Edge Induced Kőnig Subgraph is fixed-parameter tractable. To see this, suppose that (G, k) is an instance of the problem; we are to decide whether G has an edge induced Kőnig subgraph with at least k edges. Note that if the parameter $k \leq 3m/5$ then we answer yes and use the approximation algorithm described in the previous subsection to obtain an edge induced Kőnig subgraph with at least k edges. If $k > 3m/5$ then we simply use a trivial $O^*(2^m)$ brute-force algorithm to decide the question. This FPT algorithm has time complexity $O^*(2^{5k/3})$.

In this subsection, we give an $O^*(2^k)$ FPT algorithm for Edge Induced Kőnig Subgraph on connected graphs by using an exact algorithm for the optimization version of the problem. To this end, we describe an $O^*(2^n)$ algorithm for this problem using a simple structural result characterizing minimal Kőnig edge deletion sets of a graph.

Theorem 4.18. *Let $G = (V, E)$ be a graph. If E' is a minimal Kőnig edge deletion set of G then there exists $V' \subseteq V$ such that $E(G[V']) = E'$, that is, the edge set of the subgraph induced by V' is precisely E'.*

Proof. Let E' be a minimal Kőnig edge deletion set of G. Then $G' = (V, E - E')$ is Kőnig. Then the vertex set of G' can be partitioned into V_1 and V_2 such that V_2 is a maximal independent set and there exists a matching saturating V_1 that lies across the cut (V_1, V_2). Let $V' = V_2$. Since E' is minimal, it is clear that $E(G[V_2]) = E'$. This completes the proof. □

Problem	*Parameterized Complexity*	*Approximability*
KVD/ AGVC	FPT.	$O(\log n \log\log n)$ approx. algorithm; NP-hard to approximate to within 1.7212; no constant-factor approx. algorithm assuming UGC.
KED	Open.	NP-hard to approximate to within 2.88; no constant-factor approx. algorithm assuming UGC.
VKS	W[1]-hard.	no factor-$O(n^{1-\epsilon})$ approx. algorithm.
EKS	FPT.	5/3-approx. algorithm for general graphs; 4/3-approx. algorithm for graphs with a perfect matching

Table 4.1: List of problems dealt with in this chapter.

Our exact algorithm for the optimization version of Edge Induced Kőnig Subgraph simply enumerates all possible subsets $V' \subseteq V$, deletes all edges E' in $G[V']$ and checks whether $G - E'$ is Kőnig. The algorithm returns an edge set $E' = E(G[V'])$ of smallest size such that $G - E'$ is Kőnig.

Theorem 4.19. *Given an n-vertex graph $G = (V, E)$, the optimization version of the* Kőnig Edge Deletion *(and hence the optimization version of* Edge Induced Kőnig Subgraph*) can be solved in time $O^*(2^n)$ and space polynomial in n.*

Theorem 4.20. Edge Induced Kőnig Subgraph *can be solved in $O^*(2^k)$ time in connected undirected graphs.*

Proof. Let (G, k) be an instance of Edge Induced Kőnig Subgraph where G is a graph with m edges and n vertices. A connected graph has a spanning tree which, being bipartite, is Kőnig. Since a tree has $n-1$ edges, if $k \leq n-1$ we answer yes; else $n \leq k+1$ and we use Theorem 4.19 to obtain an $O^*(2^k)$ time algorithm for Edge Induced Kőnig Subgraph. □

4.6 Conclusion and Open Problems

In this chapter we introduced and studied vertex and edge versions of the Kőnig Subgraph problem from the points of view of parameterized complexity and approximation algorithms. Our results are summarized in Figure 4.1. We showed that Kőnig Vertex Deletion is in FPT whereas Vertex Induced Kőnig Subgraph is W[1]-hard. The Edge Induced Kőnig Subgraph problem is in FPT and we conjecture that Kőnig Edge Deletion is W[1]-hard. Some obvious open problems are:

1. What is the parameterized complexity of the Kőnig Edge Deletion problem?

2. Is there a better FPT-algorithm for Kőnig Vertex Deletion perhaps without making use of the algorithm for AGVC?

3. Are there better approximation algorithms for all these problems?

4. Theorem 4.15 shows that any graph G with a maximum matching M has an edge-induced subgraph of size at least

$$\frac{3|E_M|}{4} + \frac{|E - E_M|}{2} + \frac{|M|}{4}.$$

   Is this lower bound tight? What is the parameterized complexity of the above-guarantee question with respect to this lower bound?

Chapter 5

The Unique Coverage Problem

In this chapter we study the parameterized complexity of a problem known as Unique Coverage, a variant of the classic Set Cover problem. This problem is an analog of Longest Common Subsequence (see Chapter 1) in that it admits several parameterizations and we show that all, except the standard parameterization and a generalization of it, are unlikely to be fixed-parameter tractable. During our study of the standard parameterized version of Unique Coverage we will have occasion to use several techniques. We use results from extremal combinatorics to obtain the best-known kernel for Unique Coverage and the well-known color-coding technique of Alon et al. [6] to show that a weighted version of this problem is in FPT.

Our application of color-coding uses an interesting variation of s-perfect hash families called (k, s)-hash families which were studied by Alon et al. [2] in the context of a class of codes called parent identifying codes [14]. To the best of our knowledge, this is the first application of (k, s)-hash families outside the domain of coding theory. We prove the existence of such families of size smaller than the best-known s-perfect hash families using the probabilistic method [5]. Explicit constructions of such families of size promised by the probabilistic method is open.

5.1 Motivation and Known Results

The Unique Coverage problem was introduced by Demaine et al. [43] as a natural maximization version of Set Cover and has applications in several areas including wireless networks and radio broadcasting. This problem is defined as follows. Given a ground set $\mathcal{U} = \{1, \ldots, n\}$, a family of subsets $\mathcal{F} = \{S_1, \ldots, S_t\}$ of $\mathcal{U}$ and a nonnegative integer k, does there exist a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that at least k elements are covered uniquely by $\mathcal{F}'$? An element is covered uniquely by $\mathcal{F}'$ if it appears in exactly one set of $\mathcal{F}'$. The optimization version requires us to maximize the number of uniquely covered elements.

We also consider the following weighted version of Unique Coverage, called Budgeted Unique Coverage, defined as follows: Given a ground set $\mathcal{U} = \{1, \ldots, n\}$, a profit p_i for each element $i \in \mathcal{U}$, a family $\mathcal{F}$ of subsets of $\mathcal{U}$, a cost c_i for each set $S_i \in \mathcal{F}$, a budget B and a nonnegative integer k, does there exist a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with total cost at most B such that the total profit of uniquely covered elements is at least k? The optimization version asks for a subset $\mathcal{F}'$ of total cost at most B such that the total profit of uniquely covered elements is maximized.

The original motivation for this problem is a real-world application arising in wireless networks [43]. Assume that we are given a map of the densities of mobile clients along with a set of possible base stations, each with a specified building cost and a specified coverage region. The goal is to choose a set of base stations, subject to a budget on the total building cost, in order to maximize the density of served clients. The difficult aspect of this problem is the interference between base stations. A mobile client's reception is better when it is within the range of a few base stations. An ideal situation is when every mobile client is within the range of exactly one base station. This is the situation modelled by the Budgeted Unique Coverage problem. The Unique Coverage problem is closely related to a single "round" of the Radio Broadcast problem [13]. For more about this relation, see Demaine et al. [43].

One can also view the Unique Coverage problem as a generalization of the Max Cut problem [43]. The input to the Max Cut problem consists of a graph $G = (V, E)$ and the goal is to find a cut (T, T'), where $\emptyset \neq T \subset V$ and $T' = V \setminus T$, that maximizes the number of edges with one endpoint in T and the other endpoint in T'. Let $\mathcal{U}$ denote the set of edges of G and for each vertex $v \in V$ define $S_v = \{e \in E \colon e \text{ is incident to } v\}$. Finally let $\mathcal{F} = \bigcup_{v \in V} \{S_v\}$. Then G has a cut (T, T') with at least k edges across it if and only if $\mathcal{F}' = \bigcup_{v \in T} \{S_v\}$ uniquely covers at least k elements of the ground set.

Demaine et al. [43] considered the approximability of Unique Coverage. On the positive side, they give an $\Omega(1/\log n)$-approximation for Budgeted Unique Coverage. Moreover, if the ratio between the maximum cost of a set and the minimum profit of an element is bounded by b, then there exists an $\Omega(1/\log b)$-approximation algorithm for the weighted version. They show that Unique Coverage is hard to approximate to within a factor of $O(1/\log^c n)$ for some constant c depending on $\epsilon > 0$, assuming NP $\not\subseteq$ BPTIME(2^{n^ϵ}) for some ϵ. They strengthen this inapproximability to $1/(\epsilon \log n)$ for some $\epsilon > 0$ based on a hardness hypothesis for Balanced Bipartite Independent Set.

Erlebach and van Leeuwen [51] study the approximability of geometric versions of the Unique Coverage problem. Among the many versions that they consider is Unique Coverage on Unit Disks, a variant in which each set is a unit disk in $\mathbb{R}^2$, for which they give a factor-18 approxima-

tion algorithm. They also consider a variant called Unique Coverage on Disks of Bounded Ply and design an asymptotic fully polynomial-time approximation scheme (FPTAS$^{\omega}$) for it.

5.2 Results in this Chapter

As with many problems in parameterized complexity, the Unique Coverage problem can be parameterized in a number of ways. We first consider an extensive list of plausible parameterizations of the problem in Section 5.3 and discuss their parameterized complexity. Our results show that barring the standard parameterized version (where the parameter is the number of uniquely covered elements) and a generalization of it, the remaining parameterized problems are unlikely to be fixed-parameter tractable.

In Section 5.4 we consider the standard parameterized version of Unique Coverage. We show that a special case of this version where any two sets in the input family intersect in at most c elements is fixed-parameter tractable by demonstrating a polynomial kernel of size k^{c+1}. This leads to a problem kernel of size k^k for the general case, proving that Unique Coverage is fixed-parameter tractable. Then using results from extremal combinatorics on strong systems of distinct representatives we obtain a 4^k kernel. This is essentially the best possible kernel for this problem since Dom et al. have shown that Unique Coverage does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to the third level [46]. Note that the size of an instance $(\mathcal{U}, \mathcal{F}, k)$ of Unique Coverage is $|\mathcal{U}| + |\mathcal{F}|$. Therefore when we say that there exists a kernel for Unique Coverage of size $g(k)$, what we mean is that there exists a kernelization algorithm that produces an equivalent instance $(\mathcal{U}', \mathcal{F}', k')$ such that $|\mathcal{U}'| + |\mathcal{F}'| \leq g(k)$.

In Section 5.5 we consider the Budgeted Unique Coverage problem. For this problem too, there are several variants. If the profits and costs are allowed to be arbitrary positive rational numbers, then Budgeted Unique Coverage, with parameters B and k, is not fixed-parameter tractable unless P = NP. If we restrict the costs and profits to be positive integers and parameterize by B, then the problem is W[1]-hard. However if we parameterize with respect to both B and k then we show, using an application of the color-coding technique, that the problem is fixed-parameter tractable. In fact, we show that this remains true even when the costs are positive integers and profits are rationals ≥ 1 or vice versa. While derandomizing our algorithms, we use a variation of s-perfect hash families called (k, s)-hash families and show the existence of such families of size smaller than the best-known s-perfect hash families. We also modify the algorithms for Budgeted Unique Coverage to two special cases: Unique Coverage and Budgeted Max Cut.

5.3 Unique Coverage: Which Parameterization?

First consider the following parameterized problem: given $(\mathcal{U}, \mathcal{F})$, find a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ that covers all of $\mathcal{U}$, each element being covered at most k times and at least once, assuming k as parameter. This is a practical parameterization for mobile-computing applications. Unfortunately, this problem is not fixed-parameter tractable unless P = NP as the case $k = 1$ reduces to the NP-complete Exact Cover problem [64].

An alternate parameterization with a similar motivation of covering each element a small number of times is as follows: given $(\mathcal{U}, \mathcal{F})$, find a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most $|\mathcal{F}| - k$ that covers all of $\mathcal{U}$. Call this problem All But k Coverage.

We show that this problem is W[1]-hard by a fixed-parameter reduction from the W[1]-complete Red/Blue Nonblocker problem [47]:

Input:	A bipartite graph $G = (R \uplus B = V, E)$ with its vertex set partitioned into a red set R and a blue set B, and a nonnegative integer k.
Parameter:	The integer k.
Question:	Does there exist a set T of at least k red vertices such that the vertices in $R \setminus T$ dominate all vertices in B?

Theorem 5.1. All But k Coverage *is* W[1]*-hard with respect to* k *as parameter.*

Proof. Given an instance $(G = (R \uplus B, E), k)$ of Red/Blue Nonblocker, let $(\mathcal{U}, \mathcal{F}, k')$ be an instance of All But k Coverage defined as follows: $\mathcal{U} := B$, $\mathcal{F} := R$, where each red vertex is interpreted as the set of blue vertices it dominates in G, and $k' = k$. Now it is easy to see that there exist at least k red vertices such that the remaining red vertices dominate all blue vertices if and only if there exists a subfamily of $\mathcal{F}$ size at most $|\mathcal{F}| - k$ that covers all of $\mathcal{U}$. □

We also note that the following parameterized versions of Unique Coverage are unlikely to be fixed-parameter tractable. Given $(\mathcal{U}, \mathcal{F})$ and nonnegative integers k and t as parameters,

1. Does there exist a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most k that covers all of $\mathcal{U}$ with each element being covered at most t times? This version is not fixed-parameter tractable as the case $t = 1$ is W[1]-hard by a reduction from Perfect Code [28] as shown below.

2. Does there exist a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most k that covers all of $\mathcal{U}$ with each element being covered at most $|\mathcal{F}| - t$ times? This is W[2]-hard as the case $t = 0$ reduces to the Set Cover problem which is W[2]-complete.

Call the version in Item 1 above with $t = 1$ the Disjoint Set Cover problem which we now show to be W[1]-hard by a reduction from Perfect Code [28].

Perfect Code
Input: A graph $G = (V, E)$ and an integer k.
Parameter: The integer k.
Question: Does G have a k-element perfect code?

A perfect code is a vertex subset $V' \subseteq V$ such that for all $u \in V$ we have $|N[u] \cap V'| = 1$, where $N[u]$ is the closed neighborhood of vertex u, that is, every vertex is dominated by exactly one vertex in V'. Given an instance (G, k) of Perfect Code construct an instance $(\mathcal{U}, \mathcal{F}, k)$ of Disjoint Set Cover by setting $\mathcal{U} := V(G)$ and $\mathcal{F} := \{N[v] \colon v \in V\}$.

Lemma 5.1. *The graph G has a k-element perfect code if and only if there exists a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of pairwise disjoint sets and of size at most k that covers all of $\mathcal{U}$.*

Proof. Let $\{v_1, \ldots, v_k\} \subseteq V$ be a k-element perfect code of G. Then clearly $\bigcup_{i=1}^{k} N[v_i]$ covers all of $\mathcal{U}$ and the sets $N[v_1], \ldots, N[v_k]$ are pairwise disjoint. For if $x \in N[v_i] \cap N[v_j]$, $i \neq j$, then $|N[x] \cap \{v_1, \ldots, v_k\}| \geq 2$, which contradicts the definition of a perfect code. Conversely if $N[v_1], \ldots, N[v_k]$ is a collection of pairwise disjoint sets that covers all of $\mathcal{U}$ then clearly $v_1, \ldots, v_k$ is a k-element perfect code for G. □

Theorem 5.2. Disjoint Set Cover *is* W[1]*-hard with respect to the number of sets in the solution as parameter.*

Finally we consider a generalization of the standard parameterized version: Gen Unique Coverage: Given $(\mathcal{U}, \mathcal{F})$ and nonnegative integers k and t, does there exist a subfamily of $\mathcal{F}$ that covers k elements at least once and at most t times, where k is the parameter? Setting $t = 1$ gives us the standard parameterized version of Unique Coverage. We note that Gen Unique Coverage is fixed-parameter tractable as the kernel result for the standard parameterized version works for this problem also. We elaborate this further in Section 5.4.2.

5.4 Unique Coverage: The Standard Version

In this section we study the standard parameterized version of Unique Coverage. Let $(\mathcal{U} = \{1, \ldots, n\}, \mathcal{F} = \{S_1, \ldots, S_m\}, k)$ be an instance of this problem, where k is the parameter. Apply the following rules on this instance until no longer applicable.

R1 If there exists $S_i \in \mathcal{F}$ such that $|S_i| \geq k$, then the given instance is a YES-instance.

R2 If there exists $S_1, S_2 \in \mathcal{F}$ such that $S_1 = S_2$, then delete S_1.

It is easy to see that these are indeed reduction rules for UNIQUE COVERAGE. For in the first case S_i is a solution and in the second case, it is clear that no solution need have both S_1 and S_2. In the following we always assume that the given instance of UNIQUE COVERAGE is reduced with respect to the above rules.

As a warm-up, we first begin with the simple case where each element of $\mathcal{U}$ is contained in at most b sets of $\mathcal{F}$. A special case of this situation is MAX CUT where $b = 2$.

Lemma 5.2. *If each element $e \in \mathcal{U}$ occurs in at most b sets of $\mathcal{F}$ then the* UNIQUE COVERAGE *problem admits a kernel of size $b(k-1)$.*

Proof. Find a maximal collection $\mathcal{F}'$ of pairwise disjoint sets in $\mathcal{F}$. If $|\bigcup_{S \in \mathcal{F}'} S| \geq k$, we are done. Therefore assume that $|\bigcup_{S \in \mathcal{F}'} S| \leq k-1$. Since every set in $\mathcal{F} \setminus \mathcal{F}'$ intersects some set in $\mathcal{F}'$, and every element of $\mathcal{U}$ occurs in at most b sets in $\mathcal{F}$, we have $|\mathcal{F} \setminus \mathcal{F}'| \leq (k-1)(b-1)$. But $|\mathcal{F}'| \leq k-1$ and so $|\mathcal{F}| \leq b(k-1)$. □

5.4.1 Bounded Intersection Size

Now consider the situation where for all $S_i, S_j \in \mathcal{F}$, $i \neq j$, we have $|S_i \cap S_j| \leq c$, for some constant c. In this case we say that the problem instance has *intersection size bounded by c* and show that the problem admits a polynomial kernel of size $O(k^{c+1})$. First consider the case when $|S_i \cap S_j| \leq 1$.

Lemma 5.3. *Suppose that for all $S_i, S_j \in \mathcal{F}$, $i \neq j$, we have $|S_i \cap S_j| \leq 1$. If an element $e \in \mathcal{U}$ is covered by at least $k+1$ sets, then one can obtain a solution covering k elements uniquely in polynomial time.*

Proof. Suppose an element $e \in \mathcal{U}$ is covered by the sets $S_1, \ldots, S_{k+1}$. Then by reduction rule R2, at most one of these sets can have size 1. The remaining k sets uniquely cover at least one element each. □

One can now easily obtain a kernel of size k^2 for the case when the intersection size is at most 1.

Lemma 5.4. *Suppose that for all $S_i, S_j \in \mathcal{F}$, $i \neq j$, we have $|S_i \cap S_j| \leq 1$. If $|\mathcal{F}| \geq k^2$, then there exists $\mathcal{T} \subseteq \mathcal{F}$ that covers at least k elements uniquely.*

Proof. If an element appears in at least $k+1$ sets then we are done by Lemma 5.3. Otherwise every element appears in at most k sets and by Lemma 5.2 we have a kernel of size $k(k-1) < k^2$. □

Next, we generalize these observations to the case when $|S_i \cap S_j| \leq c$, for some constant c.

Theorem 5.3. *Suppose that for all $S_i, S_j \in \mathcal{F}$, $i \neq j$, we have $|S_i \cap S_j| \leq c$, for some positive constant c. If $|\mathcal{F}| \geq k^{c+1}$ then one can, in polynomial time, find a subset $\mathcal{T} \subseteq \mathcal{F}$ that covers k elements uniquely.*

Proof. By induction on c. For $c = 1$, this follows from Lemma 5.4. Assume the theorem to hold for $c \geq 1$. Let $c \geq 2$. Greedily obtain a maximal collection $\mathcal{F}' = \{S_1, \ldots, S_p\}$ of pairwise disjoint sets. If $|\bigcup_{S_i \in \mathcal{F}'} S_i| \geq k$ then we are done. Therefore assume $|\bigcup_{S \in \mathcal{F}'} S| \leq k-1$ (this also implies $p \leq k-1$). Since $|\mathcal{F}| \geq k^{c+1}$, and since every set in $\mathcal{F}$ intersects with at least one set in $\mathcal{F}'$, there exists $e \in \bigcup_{S \in \mathcal{F}'} S$ such that at least k^c+1 sets in $\mathcal{F} \backslash \{S_1, \ldots, S_p\}$ contain e. For otherwise, $|\mathcal{F}| \leq (k-1)k^c + p < k^{c+1}$, a contradiction. Let $T_1, \ldots, T_{k^c+1}$ be some $k^c + 1$ such sets. Delete e from each of these sets. We obtain at least k^c nonempty distinct sets $T'_1, \ldots, T'_{k^c}$ (there is at most one set consisting only of the element e which is deleted in this process). Note that any two of these sets intersect in at most $c-1$ elements. By induction hypothesis, there exists a collection $\mathcal{T}' \subseteq \{T'_1, \ldots, T'_{k^c}\}$ that uniquely covers at least k elements, and thus there exists a collection $\mathcal{T} \subseteq \{T_1, \ldots, T_{k^c}\}$ that uniquely covers at least k elements (just take the solution for T' and add e to every set in it). This proves the theorem. □

Corollary 5.1. Unique Coverage *admits a kernel of size k^{c+1} in the case where any two sets in the family intersect in at most c elements.*

By reduction rule R1 we have $c \leq k-1$ and therefore for the general case we have a kernel of size k^k.

Corollary 5.2. Unique Coverage *is fixed-parameter tractable and admits a problem kernel of size k^k.*

An algorithm that checks all possible subsets of a family of size k^k to see whether any of them uniquely covers at least k elements is an FPT-algorithm with time complexity $O^*(2^{(k^k)})$. But note that we can assume without loss of generality that every set in the solution covers at least one element uniquely. Thus it suffices to check whether subfamilies of size at most k uniquely cover at least k elements. This can be done in time $O^*(k^{k^2}) = O^*(2^{k^2 \log k})$. However, it turns out that a much better kernel can be obtained for the general case.

5.4.2 A Better Kernel for the General Case

We now show that Unique Coverage has a kernel of size 4^k using a result on strong systems of distinct representatives. Given a family of sets $\mathcal{F} = \{S_1, \ldots, S_m\}$, *a system of distinct representatives* for $\mathcal{F}$ is an

m-tuple $(x_1, \ldots, x_m)$ where the elements x_i are distinct and $x_i \in S_i$ for all $i = 1, 2, \ldots, m$. Such a system is *strong* if we additionally have $x_i \notin S_j$ for all $i \neq j$.

Theorem 5.4 ([83])**.** *In any family of more than $\binom{r+s}{s}$ sets of cardinality at most r, at least $s+2$ of its members have a strong system of distinct representatives.*

Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of Unique Coverage, put $r = k-1$ and $s = k$ in the statement of the above theorem and we have a kernel of size $\binom{2k-1}{k-1} \leq \binom{2k}{k} \leq 2^{2k}$.

Corollary 5.3. Unique Coverage *admits a problem kernel of size 4^k.*

As noted before, this is essentially the best possible kernel for this problem since Dom et al. have shown that Unique Coverage does not admit a kernel of size polynomial in k unless the Polynomial Hierarchy collapses to the third level [46].

Corollary 5.4. *There is an $O^*(4^{k^2})$ time algorithm for the* Unique Coverage *problem.*

Proof. A subfamily that covers k elements uniquely has size at most k. Therefore it is sufficient to consider all possible size-k subfamilies of the 4^k-kernel. This takes time $O^*(4^{k^2})$. □

In Section 5.6, we provide a better algorithm for Unique Coverage with running time $O(2^{O(k \log\log k)} \cdot mk + m^2)$, where m is the number of sets in the input family. At this point, we note that Gen Unique Coverage (see Section 5.3) is fixed-parameter tractable. Recall the problem definition: Given $(\mathcal{U}, \mathcal{F})$ and nonnegative integers k and t, does there exist a subfamily of $\mathcal{F}$ that covers k elements at least once and at most t times? Here k is the parameter. An instance $(\mathcal{U}, \mathcal{F}, k, t)$ is trivially a yes-instance if $|\mathcal{F}| > 4^k$. Otherwise $|\mathcal{F}| \leq 4^k$ and we have a kernel.

Corollary 5.5. Gen Unique Coverage *is fixed-parameter tractable.*

For the case where each set of the input family has size at most b, for some constant b, there is a better kernel. By Theorem 5.4, if there exists at least $\binom{b+k}{k}$ sets in the input family, then there exists at least k sets with a strong system of distinct representatives.

Corollary 5.6. *If each set $S \in \mathcal{F}$ has size at most b then the* Unique Coverage *problem has a kernel of size $O(2^{b+k})$.*

5.5 Budgeted Unique Coverage

In this section we consider the Budgeted Unique Coverage problem where each set in the input family has a cost and each element in the universe has a profit; the goal is to decide whether there exists a subfamily of total cost at most B that uniquely covers elements of total profit at least k. By parameterizing on k or B or both we obtain different parameterized versions of this decision question.

5.5.1 Intractable Parameterized Versions

We first consider the Budgeted Max Cut problem which is a specialization of the Budgeted Unique Coverage problem. An instance of this problem is an undirected graph $G = (V, E)$ with a cost function $c\colon V \to \mathbb{Q}^+$ on the vertex set and a profit function $p\colon E \to \mathbb{Q}^+$ on the edge set; positive rational numbers B and k. The question is whether there exists a cut (T, T') such that the total cost of the vertices in T is at most B and the total profit of the edges crossing the cut is at least k.

We first show that the Budgeted Max Cut problem with arbitrary positive rational costs and profits is probably not in FPT.

Lemma 5.5. *The* Budgeted Max Cut *problem with arbitrary positive rational costs and profits with parameters B and k is not in* FPT*, unless* P = NP*.*

Proof. Suppose there exists an algorithm for Budgeted Max Cut (with arbitrary positive costs and profits) with running time $O(f(k, B) \cdot p(n))$, where p is a polynomial in n. We will use this to solve the decision version of Max Cut in polynomial time. Let $(G = (V, E), k)$ be an instance of the Max Cut problem, where $|V| = n$. Assign each vertex of the input graph cost $1/n$ and each edge profit $1/k$. Let the budget $B = 1/2$ and the profit $k' = 1$. Clearly, G has a maximum cut of size at least k if and only if there exists $S \subseteq V$ of total cost at most B such that the total profit of the edges crossing the cut $(S, V \setminus S)$ is at least k'. And this can be answered in time $O(f(1, 1/2) \cdot p(|V|))$, implying P = NP. □

Theorem 5.5. *The* Budgeted Unique Coverage *problem with arbitrary positive rational costs and profits is not in* FPT*, unless* P = NP*.*

We next show that the Budgeted Max Cut problem parameterized by the budget B alone is W[1]-hard even when the costs and profits are positive integers.

Lemma 5.6. *The* Budgeted Max Cut *problem with positive integer costs and profits, parameterized by the budget B is* W[1]*-hard.*

Proof. To show W[1]-hardness, we exhibit a fixed-parameter reduction from the Independent Set problem to the Budgeted Max Cut problem with unit costs and profits. Let $(G = (V, E), B)$ be an instance of Independent Set with $|V| = n$. For every vertex $u \in V$ add $|V| - 1 - \deg(u)$ new vertices and connect them to u. Call the resulting graph G'. Define $c(v) = 1$ for all $v \in V(G')$ and $p(e) = 1$ for all $e \in E(G')$. Note that every vertex $u \in G$ has degree $|V| - 1$ in G'. We let $(G' = (V', E'), B, k = B(n-1))$ be the instance of Budgeted Max Cut.

Claim. The graph G has an independent set of size B if and only if G' has a cut $(S, V' \setminus S)$ such that $|S| = B$ and at least $k = B(n-1)$ edges lie across it.

If G has an independent set S of size B, then clearly S is independent in G'. The cut $(S, V' \setminus S)$ does indeed have $B(n-1)$ edges crossing it, as every vertex of S has degree $n-1$. Next suppose that G' has a cut $(S, V' \setminus S)$ with $B(n-1)$ edges crossing it such that $|S| = B$. Note that every vertex in S must be a vertex from G. Otherwise the cut cannot have $B(n-1)$ edges crossing it. Suppose two vertices u and v in S are adjacent. Then both u and v contribute less than $n-1$ edges to the cut. Since each vertex in S contributes at most $n-1$ edges to the cut, the number of edges crossing the cut must be less than $B(n-1)$, a contradiction. Hence S is independent in G' and and G has an independent set of size B. □

Since the Budgeted Unique Coverage problem is a generalization of Budgeted Max Cut we have

Theorem 5.6. *The* Budgeted Unique Coverage *problem with positive integer costs and profits, parameterized by the budget B is* W[1]*-hard.*

5.5.2 Parameterizing by both B and k

We now assume that, unless otherwise mentioned, both costs and profits are positive integers and that both B and k are parameters. Let $(\mathcal{U}, \mathcal{F}, c, p, B, k)$ be an instance of Budgeted Unique Coverage. We may assume that for all $S_i, S_j \in \mathcal{F}$, $i \neq j$, we have

1. $S_i \neq S_j$;
2. $c(S_i) \leq B$;
3. $p(S_i) \leq k - 1$;
4. $B \geq 2$.

For if $c(S_i) > B$ then S_i cannot be part of any solution and may be discarded; if $p(S_i) \geq k$ then the given instance is trivially a yes-instance. Note that

the condition $p(S_i) \leq k-1$ implies that $|S_i| \leq k-1$. Also observe that if the first three conditions hold, then in a YES-instance we must have $B \geq 2$.

For some of the results in this section, we need the following lemma.

Lemma 5.7. *For all $t \geq 2k$, $\left(\frac{t-k}{t}\right)^t \geq (2e)^{-k}$.*

Proof. We show that $\left(\frac{t}{t-k}\right)^t \leq (2e)^k$ for all $t \geq 2k$.

$$\begin{aligned}\left(\frac{t}{t-k}\right)^t &= \left(1+\frac{k}{t-k}\right)^{t-k} \cdot \left(1+\frac{k}{t-k}\right)^k \\ &\leq e^k \cdot \left(1+\frac{k}{t-k}\right)^k \\ &\leq e^k \cdot 2^k = (2e)^k\end{aligned}$$

The last inequality follows since $t \geq 2k$. □

Demaine et al. [43] show that there exists an $\Omega(1/\log n)$-approximation algorithm for Budgeted Unique Coverage (Theorem 4.1). We use the same proof technique to show the following.

Lemma 5.8. *Let $(\mathcal{U}, \mathcal{F}, c, p, B, k)$ be an instance of* Budgeted Unique Coverage *and let $c\colon \mathcal{F} \to \mathbb{Q}^{\geq 1}$ and $p\colon \mathcal{U} \to \mathbb{Q}^{\geq 1}$. Then either*

1. *one can find, in polynomial time, a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with total cost at most B such that the total profit of elements uniquely covered by $\mathcal{F}'$ is at least k; or*

2. *for every subfamily $\mathcal{H}$ with total cost at most B, we have $|\bigcup_{S\in\mathcal{H}} S| \leq 18k \log B$.*

Proof. Let $\mathcal{H} = \{S_1, \ldots, S_r\}$ be a subfamily of $\mathcal{F}$ with budget at most B which maximizes $|\mathcal{U}'|$, where $\mathcal{U}' = \bigcup_{S\in\mathcal{H}} S$. For $u \in \mathcal{U}$, let f_u denote the number of sets of $\mathcal{H}$ containing u. Partition $\mathcal{U}'$ into sets $C_0, C_1 \ldots C_{t-1}$, such that $u \in C_i$ if $2^i \leq f_u \leq 2^{i+1}-1$. Note that i ranges from 0 to $\log(r+1)-1$, since the frequency of any element in C_{t-1} is at most $2^t - 1$, which in turn is at most r. So $t \leq \log(r+1)$.

Clearly there exists j such that $|C_j| \geq |\mathcal{U}'|/\log(r+1)$. Fix j to be an index for which $|C_j| \geq |\mathcal{U}'|/\log(r+1)$. Fix $u \in C_j$ and note that $2^j \leq f_u \leq 2^{j+1}-1$. Construct a subfamily $\mathcal{F}'$ from $\mathcal{H}$ by going through each set in $\mathcal{H}$ and including it in $\mathcal{F}'$ with probability $1/2^{j+1}$. Denote by l_u

the probability that u is covered uniquely by $\mathcal{F}'$. Then

$$\begin{aligned} l_u &= \left(\tfrac{f_u}{2^{j+1}}\right)\left(1-\tfrac{1}{2^{j+1}}\right)^{(f_u-1)} \\ &\geq \tfrac{1}{2}\left(1-\tfrac{1}{2^{j+1}}\right)^{(f_u-1)} && \text{(since } f_u \geq 2^j\text{)}. \\ &\geq \tfrac{1}{2}\left(1-\tfrac{1}{2^{j+1}}\right)^{(2^{j+1}-2)} && \text{(since } f_u \leq 2^{j+1}-1\text{)}. \\ &\geq \tfrac{1}{2}\left(1-\tfrac{1}{2^{j+1}}\right)^{(2^{j+1})} \\ &\geq \tfrac{1}{4e}. && \text{(by Lemma 5.7)}. \end{aligned}$$

Let X_u be an indicator random variable which takes value 1 if u is covered uniquely by the subfamily $\mathcal{F}'$, and 0 otherwise. Also, let $X = \sum_u X_u$. Then

$$E(X) = \sum_u E(X_u) \geq \sum_{u \in C_j} E(X_u) = \sum_{u \in C_j} l_u \geq \frac{|\mathcal{U}'|}{4e \log(r+1)}.$$

Let the set of elements uniquely covered by $\mathcal{F}'$ be Q. Then the total profit of these uniquely covered elements is at least $|Q|$ and

$$E(|Q|) \geq \frac{1}{4e} \cdot \frac{|\mathcal{U}'|}{\log(r+1)},$$

as every uniquely covered element contributes at least 1 to the total profit. Since r is bounded above by B, the total expected profit of these elements is at least

$$\frac{1}{4e} \cdot \frac{|\mathcal{U}'|}{\log(B+1)}. \tag{5.1}$$

This implies that if k is at most the quantity in expression 5.1 then the total profit of uniquely covered elements of subfamily $\mathcal{F}'$ is at least k; else, we have $|\mathcal{U}'| \leq 4ek \log(B+1) \leq 12k \log B$.

We can design an algorithm that finds a subfamily of total budget at most B that uniquely covers elements with total profit at least k as follows. Define p' to be the unit profit function, that is, $p'(u) = 1$ for all $u \in \mathcal{U}$. We find a subfamily $\mathcal{H}'$ in place of $\mathcal{H}$ using an approximation algorithm for the Maximum Coverage problem (where the objective is to maximize the elements covered by the subfamily). To do this we run the polynomial-time $(1-1/e)$-ratio approximate algorithm described in [89] for the Budgeted Maximum Coverage problem with universe $\mathcal{U}$, family $\mathcal{F}$, cost function c, profit function p' and budget B. This returns a subfamily $\mathcal{H}'$ of total cost at most B that covers at least $(1-1/e) \cdot \text{OPT} = d \cdot \text{OPT}$ elements, where OPT denotes the maximum number of elements in any family of total cost at most B. From $\mathcal{H}'$ one can deterministically obtain a subfamily $\mathcal{F}'$ that uniquely covers elements with total profit at least

$$\frac{1}{4e} \cdot \frac{d \cdot \text{OPT}}{\log(B+1)}$$

by the method of conditional probabilities (see [104]).

Hence depending on the value of k one can, in polynomial time, either obtain a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with budget at most B such that the total profit of uniquely covered elements is at least k; or every subfamily $\mathcal{H}$ with budget at most B contains at most $\frac{4e}{1-1/e} \cdot k \log(B+1) \leq 18k \log B$ elements. □

The first step of our algorithm is to apply Step 1 of Lemma 5.8. Therefore from now on we assume that every subfamily of total cost at most B covers at most $18k \log B$ elements of the universe.

We now proceed to show that Budgeted Unique Coverage is in FPT by an application of the color-coding technique. We first show this for the case when the costs and profits are all one and then handle the more general case of integral costs and profits. Therefore let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of Budgeted Unique Coverage with unit costs and profits. For this version of the problem, we have to decide whether there exists a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most B that uniquely covers at least k elements. A subfamily $\mathcal{F}'$ of size at most B that uniquely covers at least k elements is called a *solution subfamily*.

To develop our color-coding algorithm, we use two sets of colors $\mathcal{C}_g$ and $\mathcal{C}_b$ with the understanding that the (good) colors from $\mathcal{C}_g$ are used for the elements that are uniquely covered and the (bad) colors from $\mathcal{C}_b$ are used for the remaining elements. In the present setting, $\mathcal{C}_g = \{1, \ldots, k\}$ and $\mathcal{C}_b = \{k+1\}$. Note that for our algorithms, any subset of k colors can play the role of good colors. For ease of presentation, we fix a set of good and bad colors while describing our randomized algorithms. Our derandomized algorithms assume that any set of k colors may be good colors.

We now describe the notion of a *good configuration*. Given a function $h\colon\ \mathcal{U} \to \mathcal{C}_g \uplus \mathcal{C}_b$ and $\mathcal{F}' \subseteq \mathcal{F}$, define $\mathcal{U}(\mathcal{F}')$ to be the set of elements covered (not necessarily uniquely) by $\mathcal{F}'$ and $h(\mathcal{F}')$ to be the set of colors assigned to the elements in $\mathcal{U}(\mathcal{F}')$. That is,

$$\mathcal{U}(\mathcal{F}') := \bigcup_{S \in \mathcal{F}'} S; \quad h(\mathcal{F}') := \bigcup_{i \in \mathcal{U}(\mathcal{F}')} \{h(i)\}.$$

Definition 5.1. Given $h\colon\ \mathcal{U} \to \mathcal{C}_g \uplus \mathcal{C}_b$ and $\mathcal{C}'_g \subseteq \mathcal{C}_g$, we say that $\mathcal{F}' \subseteq \mathcal{F}$ *has a good configuration with respect to (w.r.t.) h and $\mathcal{C}'_g$* if

1. $h(\mathcal{F}') \cap \mathcal{C}_g = \mathcal{C}'_g$, and
2. there are exactly $|\mathcal{C}'_g|$ elements in $\mathcal{F}'$ that are assigned colors from $\mathcal{C}'_g$ and these elements are uniquely covered by $\mathcal{F}'$.

We also say that $\mathcal{F}$ *has a good configuration w.r.t. h and $\mathcal{C}'_g$* if there exists a subfamily $\mathcal{F}'$ with a good configuration w.r.t. h and $\mathcal{C}'_g$. Call $\mathcal{F}'$ a *witness subfamily*.

A solution subfamily (for the unit costs and profits version) is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with at most B sets and which uniquely covers at least k elements.

The next lemma shows that if $(\mathcal{U}, \mathcal{F}, B, k)$ is a YES-instance of BUDGETED UNIQUE COVERAGE with unit costs and profits, and h is chosen uniformly at random from the space of all functions from $\mathcal{U}$ to $[k+1]$, then a solution subfamily $\mathcal{F}'$ has a good configuration w.r.t. h and $\mathcal{C}_g$ with high probability. Note that such a uniformly chosen h maps every element from $\mathcal{U}$ uniformly at random to an element in $[k+1]$.

Lemma 5.9. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be a* YES*-instance of* BUDGETED UNIQUE COVERAGE *with unit costs and profits and let $h\colon \mathcal{U} \to [k+1]$ be a function chosen uniformly at random. Then a solution subfamily $\mathcal{F}'$ has a good configuration w.r.t. h and $\mathcal{C}_g$ with probability at least $2^{-k(18 \log B \log(k+1) - \log k + \log e)}$.*

Proof. Let $\mathcal{F}'$ be a solution subfamily with at most B sets that covers the elements $Q = \{i_1, \ldots, i_k\}$ uniquely. Then $p := |\mathcal{U}(\mathcal{F}')| \leq 18k \log B$. To complete the proof, we show that $\mathcal{F}'$ has a good configuration with respect to h and $\mathcal{C}_g$ with probability at least $2^{-k(18 \log B \log(k+1) - \log k + \log e)}$. For $\mathcal{F}'$ to have a good configuration, we must have $h(i) = k+1$ for all $i \in \mathcal{U}(\mathcal{F}') \setminus Q$ and $h(i_1), \ldots, h(i_k)$ a permutation of $1, \ldots, k$. The probability $\mathbf{Pr}$ that this happens is:

$$\begin{aligned}
\mathbf{Pr} &= \frac{1}{(k+1)^{|\mathcal{U}(\mathcal{F}') \setminus Q|}} \times \frac{k!}{(k+1)^k} \\
&\geq \left(\frac{k}{e}\right)^k \frac{1}{(k+1)^p} = e^{k \ln(k/e) - p \ln(k+1)} \\
&\geq e^{k \ln(k/e) - 18k \log B \ln(k+1)} \\
&= 2^{-k(18 \log B \log(k+1) - \log k + \log e)}
\end{aligned}$$

This proves the lemma. □

Given a coloring h, how do we find out whether $\mathcal{F}$ has a good configuration w.r.t. h and $\mathcal{C}_g$? We answer this next.

Finding a good configuration. Observe that if $\mathcal{F}$ has a good configuration w.r.t. h and $\mathcal{C}_g$, then any witness subfamily $\mathcal{F}'$ covers at least k elements uniquely. To locate such a family of size at most B we use dynamic programming over subsets of $\mathcal{C}_g$. To this end, let W be a $2^k \times B$ array where we identify the rows of W with subsets of $\mathcal{C}_g$ and the columns with the size of a subfamily. For a fixed coloring function h, a subset $\mathcal{C}'_g \subseteq \mathcal{C}_g$ and $1 \leq i \leq B$, define $W[\mathcal{C}'_g][i]$ as follows:

$$W[\mathcal{C}'_g][i] = \begin{cases} 1, & \text{if there exists } \mathcal{F}' \subseteq \mathcal{F}, \text{ with } |\mathcal{F}'| \leq i, \text{ with a} \\ & \text{good configuration w.r.t. } \mathcal{C}'_g \text{ and } h. \\ 0, & \text{otherwise.} \end{cases}$$

The entry corresponding to $W[\emptyset][i]$ is set to 1 for all $1 \leq i \leq B$, as a convention. We fill this array in increasing order of the sizes of subsets of $\mathcal{C}_g$. Let $\mathcal{T}$ be the family of all sets $S \in \mathcal{F}$ such that $g(S) := h(S) \cap \mathcal{C}_g \subseteq \mathcal{C}'_g$ and for each $c \in g(S)$ there exists exactly one element $e \in S$ with $h(e) = c$. Then

$$W[\mathcal{C}'_g][i] = \bigvee_{S \in \mathcal{T}} W[\mathcal{C}'_g \setminus g(S)][i-1].$$

The correctness of the algorithm is immediate. Clearly if $W[\mathcal{C}_g][B] = 1$, then a subfamily with at most B sets that uniquely covers at least k elements exists, and can be found out by simply storing the witness families $\mathcal{F}'$ for every entry in the table and backtracking. The time taken by the algorithm is $O(2^k Bmk)$, since the size of the array is $2^k B$ and each entry of the array can be filled in time $O(mk)$, where $m = |\mathcal{F}|$.

Lemma 5.10. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* Budgeted Unique Coverage *with unit costs and profits and $h\colon \mathcal{U} \to \mathcal{C}$ a coloring function. Then one can find a subfamily $\mathcal{F}'$ of size at most B with a good configuration w.r.t. h and $\mathcal{C}_g$, if there exists one, in time $O(2^k Bmk)$.*

A randomized algorithm for Budgeted Unique Coverage with unit costs and profits is as follows.

1. Randomly choose a coloring function $h\colon \mathcal{U} \to \{1, \ldots, k+1\}$.
2. Apply Lemma 5.10 and check whether there exists a family $\mathcal{F}'$ of size at most B that is witness to a good configuration w.r.t. h and $\mathcal{C}_g$. If such a family exists, return yes, else go to Step 1.

By Lemma 5.9, if the given instance is a yes-instance, the probability that a solution subfamily $\mathcal{F}'$ has a good configuration w.r.t. a randomly chosen function $h\colon \mathcal{U} \to \mathcal{C}$ and $\mathcal{C}_g$ is at least $2^{-k(18 \log B \log(k+1) - \log k + \log e)}$. By Lemma 5.10, one can find such a subfamily in time $O(2^k Bmk)$.

Theorem 5.7. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* Budgeted Unique Coverage *with unit costs and profits. There exists a randomized algorithm that finds a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in expected $O(2^{18k \log B \log(k+1)} \cdot Bmk)$ time.*

Improving the running time

It is clear that if a solution subfamily $\mathcal{F}'$ is to have a good configuration w.r.t. a randomly chosen coloring function h and $\mathcal{C}_g$, then h must assign all the non-uniquely covered elements of $\mathcal{F}'$ the color in $\mathcal{C}_b$. Intuitively, if we increase the number of colors in $\mathcal{C}_b$, we increase the probability that a specific target subfamily has a good configuration w.r.t. a randomly chosen coloring function. We formalize this intuition below.

Lemma 5.11. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be a* YES*-instance of* Budgeted Unique Coverage *with unit costs and profits; let $\mathcal{C}_g = [k]$, $\mathcal{C}_b = \{k+1, \ldots, q\}$ and $\mathcal{C} = [q]$ so that $q \geq 2k$. If $h\colon\ \mathcal{U} \to \mathcal{C}$ is chosen uniformly at random then every solution subfamily $\mathcal{F}'$ with p elements of the universe has a good configuration w.r.t. h and $\mathcal{C}_g$ with probability at least $e^{-k}\left(\frac{k}{q-k}\right)^k (2e)^{-\frac{kp}{q}}$.*

Proof. Let the set of elements uniquely covered by $\mathcal{F}'$ be $Q = \{i_1, \ldots, i_k\}$. For $\mathcal{F}'$ to have a good configuration, the function h must map every element of $\mathcal{U}(\mathcal{F}') \setminus Q$ to $\mathcal{C}_b$ and map Q to $\mathcal{C}_g$ injectively. Therefore the probability $\mathbf{Pr}$ that $\mathcal{F}'$ has a good configuration w.r.t. $\mathcal{C}_g$ and a randomly chosen h is:

$$\begin{aligned}
\mathbf{Pr} &= \frac{(q-k)^{p-k}}{q^{p-k}} \times \frac{k!}{q^k} \\
&\geq \left(\frac{q-k}{q}\right)^p \cdot \left(\frac{1}{q-k}\right)^k \cdot k^k e^{-k} \\
&\geq e^{-k} \cdot \left(\frac{k}{q-k}\right)^k \cdot \left(1 - \frac{k}{q}\right)^p \\
&\geq e^{-k} \cdot \left(\frac{k}{q-k}\right)^k \cdot (2e)^{-\frac{kp}{q}} \qquad \text{(by Lemma 5.7).}
\end{aligned}$$

This proves the lemma. □

If $(\mathcal{U}, \mathcal{F}, B, k)$ is a YES-instance of Budgeted Unique Coverage with unit costs and profits then $p \leq 18k \log B$. As we observed earlier, we have $B \geq 2$. By Lemma 5.11, a solution subfamily $\mathcal{F}'$ has a good configuration w.r.t. a randomly chosen coloring function h and $\mathcal{C}_g$ with probability at least $e^{-k} \cdot (\frac{q-k}{k})^{-k} \cdot (2e)^{-kp/q}$. Setting $q = k + p$, this expression works out to be:

$$\begin{aligned}
\mathbf{Pr} &\geq e^{-k} \cdot \left(\frac{k}{p}\right)^k \cdot (2e)^{-kp/(k+p)} \\
&\geq 2^{-k \log e - k \log \frac{p}{k} - \frac{kp}{k+p} \cdot \log 2e} \\
&= 2^{-k \log e - k \log \frac{p}{k} - \frac{k}{1+k/p} \cdot \log 2e} \\
&\geq 2^{-k \log e - k \log \frac{p}{k} - k \log 2e}.
\end{aligned}$$

Now setting $p = 18k \log B$ this expression works out to $2^{-8.1k - k \log \log B}$. Combining this with Lemma 5.10, we obtain:

Theorem 5.8. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* Budgeted Unique Coverage *with unit costs and profits. Then one can find a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in $O(2^{9.1k + k \log \log B} \cdot Bmk)$ expected time.*

Derandomization

We now discuss how to derandomize the algorithms described in the last subsection. In general, randomized algorithms based on the color-coding method are derandomized using a suitable family of hash functions or "universal sets". Fix integers k, s, t, n, where $[n]$ is the universe, $[t]$ is the set of colors, s and k denote, respectively, the size of a solution subfamily and the number of elements uniquely covered by it. Note that $t \geq k + 1$ and, by Lemma 5.8, $s = \lceil 18k \log B \rceil$. We need a family of functions from $\mathcal{U} = [n]$ to $[t]$ such that for all $S \subseteq \mathcal{U}$ of size s and all $X \subseteq S$ of size k, there exists a function h in the family which maps X injectively and the colors it assigns to the elements in $S \setminus X$ are different from the ones it assigns to those in X. See Figure 5.1

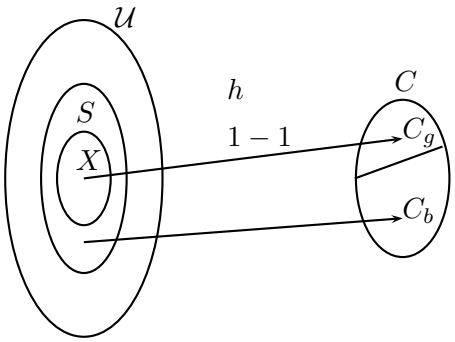


Figure 5.1: The sets in the definition of a (k, s)-hash family. We assume $\mathcal{U} = [n]$, $\mathcal{C} = [t]$, $|S| = s$ and $|X| = k$.

Such hash families are called (k, s)-hash families (with domain $[n]$ and range $[t]$) and they were introduced by Barg et al. [14] in the context of particular class of codes called parent identifying codes. At this point, we recall the definition of an (n, t, s)-*perfect hash family*.

Definition 5.2 ((n, t, s)-Perfect Hash Family)**.** A family $\mathcal{H}$ of functions from $[n]$ to $[t]$ is called an (n, t, s)-perfect hash family if for every subset $X \subseteq [n]$ of size s, there is a function $h \in \mathcal{H}$ that maps X injectively.

Note that an (n, t, s)-perfect hash family is a (k, s)-hash family with domain $[n]$ and range $[t]$ for any $k \leq s$, and a (k, s)-hash family with domain $[n]$ and range $[t]$ is an (n, t, k)-perfect hash family. Therefore (k, s)-hash families may be thought of as standing in between (n, t, k)-perfect and (n, t, s)-perfect hash families.

Our deterministic algorithm simply uses functions from these families $\mathcal{H}$ for coloring and is described in Figure 5.2. Given an instance $(\mathcal{U}, \mathcal{F}, B, k)$ of Budgeted Unique Coverage with unit costs and profits, we let $n = |\mathcal{U}|$, $\mathcal{C} = [t]$, and s to be the closest integer to our estimate in Lemma 5.8,

for each $h \in \mathcal{H}$ **do**
  for each subset $X \subseteq \mathcal{C} = [t]$ of size k **do**

1. Define $\mathcal{C}_g = X$ and $\mathcal{C}_b = \mathcal{C} \setminus X$;
2. Apply Lemma 5.10 and check whether there exists a subfamily $\mathcal{F}'$ of size at most B which has a good configuration w.r.t. $\mathcal{C}_g$ and h;
3. **if** yes, then **return** the corresponding $\mathcal{F}'$;

return NO;

Figure 5.2: Deterministic algorithm for BUDGETED UNIQUE COVERAGE.

which is $O(k \log B)$. The correctness of the algorithm follows from the description—if a witness subfamily for the given $\mathcal{F}$ exists, at least one $h \in \mathcal{H}$ will color all the uniquely covered elements of the witness subfamily distinctly, thereby resulting in a good configuration. The running time of the algorithm is $O\left(|\mathcal{H}| \cdot \binom{t}{k} \cdot 2^k Bmk\right)$.

Alon et al. [2] provide explicit constructions of (k, s)-hash families when the range is $k + 1$ and ks, respectively.

Theorem 5.9 (Alon et al. [2])**.** *Let $2 \leq k < s$. There is an explicit construction of a (k, s)-hash family $\mathcal{H}$ with domain $[n]$ and*

- *range $[k + 1]$ of size at most $2^{ck \log s} \cdot \log_{k+1} n$, for some absolute constant $c > 0$;*
- *range $[ks]$ of size $O(k^2 s^2 \log n)$.*

If $t = k + 1$, then by the above theorem, the running time of our deterministic algorithm is $O(2^{O(k \log k + k \log \log B)} \cdot Bmk \cdot \log n)$; when $t = ks$, the running time works out to be $O(2^{O(k \log k + k \log \log B)} \cdot mk^5 \cdot B \log^2 B \cdot \log n)$.

Theorem 5.10. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ an instance of the* BUDGETED UNIQUE COVERAGE *problem with unit costs and profits. Then one can find a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in time $O(2^{O(k \log k + k \log \log B)} \cdot Bmk \cdot \log n)$.*

We next give alternate running time bounds using standard (n, t, s)-perfect hash families for derandomizing our algorithm.

Theorem 5.11 ([6, 119, 32])**.** *There exist explicit constructions of (n, t, s)-perfect hash families of size*

- *$2^{O(s)} \log n$ when $t = s$; and*
- *$s^{O(1)} \log n$ when $t = s^2$.*

In fact, when $t = s$, an explicit construction of an s-perfect hash family of size $6.4^s \log^2 n$ in time $6.4^s n \log^2 n$ is known.

For $t = s$, using the construction of s-perfect hash families by Chen et al. [32], we obtain a running time of $O(6.4^s \log^2 n \cdot \binom{s}{k} \cdot 2^k \cdot Bkm)$. Since $s = O(k \log B)$, this expression simplifies to $O(2^{O(k \log B)} \cdot \log^2 n \cdot Bmk)$. For $t = s^2$, we can use a hash family of size $s^{O(1)} \log n$ [6], and the expression for the running time then works out to be $O(2^{O(k \log k + k \log\log B)} \cdot \log n \cdot Bmk)$. We thus have

Theorem 5.12. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of the* Budgeted Unique Coverage *problem with unit costs and profits. Then one can find a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in time $O(\min\{2^{O(k \log B)}, 2^{O(k \log k + k \log\log B)}\} \cdot \log^2 n \cdot Bmk)$.*

In general, there is no relation between the parameters k and B, but if B is small or nearly equal to k then the running time in Theorem 5.12 is better than that in Theorem 5.10.

We now consider existential results concerning hash families. The following is known about (n, t, s)-hash families.

Theorem 5.13 ([101])**.** *For all positive integers $n \geq t \geq s \geq 2$, there exists an (n, t, s)-perfect hash family $\Delta(n, t, s)$ of size $e^{s^2/t} s \ln n$.*

Alon et al. [2] provide existential bounds for (k, s)-hash families when the range $t = k + 1$ and one such bound (see Theorem 3 in [2]) is the following. If $\mathcal{H}$ is a (k, s)-hash family with a domain $[n]$ and range $[k + 1]$ then

$$\frac{\log_{k+1} n}{|\mathcal{H}|} = \frac{k!(s-k)^{s-k}}{s^s (s-1) \ln(k+1)} - o(1),$$

which implies that

$$|\mathcal{H}| \leq \frac{s^s (s-1) \ln(k+1) \log_{k+1} n}{k!(s-k)^{s-k}}.$$

If we assume that $s \geq 2k$, then using Lemma 5.7, this expression can be bounded above by $(2e)^k \cdot s^{k+1} \ln(k+1) \log_{k+1} n$. As stated in Theorem 5.9, Alon et al. [2] describe explicit constructions of such (k, s)-hash family of size $s^{ck} \cdot \log_{k+1} n$ which is still exponentially larger than the existential bound.

In the lemmas that follow, we provide existential bounds for an arbitrary range.

Lemma 5.12. *Let $k \leq s \leq n$ be positive integers and let $t \geq 2k$ be an integer. There exists a (k, s)-hash family $\mathcal{H}$ with domain $[n]$ and range $[t]$ of size $e^k (2e)^{sk/t} \cdot s \log n$.*

Proof. Let $\mathcal{A} = \{h\colon [n] \to [t]\}$ be the set of all functions from $[n]$ to $[t]$. For $h \in \mathcal{A}$, $S \subseteq [n]$ of size s and $X \subseteq S$ of size k, define h to be an (X,S)-*hash function* if h maps X injectively such that $h(X) \cap h(S \setminus X) = \emptyset$ and *not an* (X,S)*-hash function* otherwise.

Fix $S \subseteq [n]$ of size s and $X \subseteq S$ of size k. The probability **Pr** that a function h picked uniformly at random from $\mathcal{A}$ is an (X,S)-hash function, is given by:

$$\begin{aligned}
\mathbf{Pr} &= \frac{\binom{t}{k}k!(t-k)^{s-k}}{t^s} \\
&> \left(\frac{t}{k}\right)^k \cdot \left(\frac{k}{e}\right)^k \cdot \frac{1}{t^k} \cdot \left(\frac{t-k}{t}\right)^{s-k} \\
&= \frac{1}{e^k}\left(\frac{t-k}{t}\right)^{s-k} \\
&\geq \frac{1}{e^k} \cdot \left(\frac{1}{2e}\right)^{k(s-k)/t} \qquad \text{(By Lemma 5.7.)} \\
&\geq \frac{1}{e^k} \cdot \left(\frac{1}{2e}\right)^{ks/t}.
\end{aligned}$$

The probability that the function h is not an (X,S)-hash function is less than $1 - e^{-k}(2e)^{-ks/t}$. If we pick N functions uniformly at random from $\mathcal{A}$ then the probability that none of these functions is an (X,S)-hash function is less than $(1 - e^{-k}(2e)^{-ks/t)})^N$. The probability that none of these N functions is an (X,S)-hash function for some (S,X) pair is less than

$$\binom{n}{s}\binom{s}{k}(1 - e^{-k}(2e)^{-ks/t})^N,$$

which in turn is less than $n^s(1 - e^{-k}(2e)^{-ks/t})^N$. For this family of N functions to contain an (X,S)-hash function for every (S,X) pair, we would want

$$n^s(1 - e^{-k}(2e)^{-ks/t})^N \leq 1. \tag{5.2}$$

Inequality 5.2 implies that

$$s \ln n + N \ln(1 - e^{-k}(2e)^{-ks/t}) \leq 0.$$

One can show that given any $\epsilon \in (0,1)$ there exists $c_\epsilon > 0$ such that for all $x \in (0,\epsilon)$, we have $-c_\epsilon x \leq \ln(1-x)$. In fact, one may choose $c_\epsilon = (|\ln(1-\epsilon)| + 1)/\epsilon$. For k and s sufficiently large we have $e^{-k}(2e)^{-ks/t} < 1/4$ and we may choose $\epsilon = 1/4$. Then $c_\epsilon \approx 5.2$ and we have:

$$s \ln n - 5.2N \cdot e^{-k}(2e)^{-sk/t} \leq 0.$$

which shows that $N \geq e^k(2e)^{ks/t} \cdot s \log n$. □

Lemma 5.13. *Let $k \leq s \leq n$ be positive integers and let $t \geq k+1$. Then there exists a (k,s)-hash family with domain $[n]$ and range $[t]$ of size $2^{O(k\log(s/k))} \cdot s \log n$.*

Proof. Let $F = \Delta(n, m, s)$, the (n, m, s)-perfect hash family obtained from Theorem 5.13, where we set $m = \lceil s^2/(k\log(s/k))\rceil$. Let G be a family of functions g_X from $[m]$ to $[t]$, indexed by k-element subsets X of $[m]$ as follows. The function g_X maps X in an one-one, onto fashion to $\{1,\ldots,k\}$ and maps an element of $[m] \setminus X$ to an arbitrary element in $\{k+1,\ldots,t\}$. Our required family T of functions from $[n]$ to $[t]$ is obtained by composing the families F and G. It is easy to see that T is an (k,s)-hash family and has the claimed bound for its size. □

Note that Lemma 5.12 requires that $t \geq 2k$ and that for Lemma 5.13 we have no restriction on t. Also note that the bound in Lemma 5.13 is existential as it uses the existential bound of Theorem 5.13. If we had an explicit construction of a (k,s)-hash family satisfying the bound in Lemma 5.12, then by setting $s = t = O(k\log B)$, we would have obtained a running time of $O(2^{O(k\log\log B)} \cdot k\log B\log n)$ which is significantly better than that given in Theorem 5.12. We believe that this is motivation for studying explicit constructions of (k,s)-hash families for an arbitrary range.

Generalized Costs and Profits

Observe that the algorithms for Budgeted Unique Coverage with unit costs and profits had two components—one component ensured that the probability that a solution subfamily has a good configuration with respect to a random coloring h and $\mathcal{C}_g$ is sufficiently high (see Lemma 5.9), and the other had to do with finding a witness subfamily given a coloring (see Lemma 5.10). Note that Lemma 5.8 and the discussions proceeding Theorem 5.8 continue to hold when either the cost or the profit is integral. The derandomization procedures given in the last subsection also go through for these general cases. We now show how to modify the dynamic programming algorithm when the costs and profits are arbitrary integers.

Dynamic programming with integral costs and profits Consider an instance $(\mathcal{U},\mathcal{F},B,k)$ of the problem with cost function c and profit function p. Recall that the colors used are from $\mathcal{C} = \mathcal{C}_g \cup \mathcal{C}_b$ and that we are given a coloring function h. As before, we define an array W of size $2^k \times B$ and associate the rows with subsets of $\mathcal{C}_g$ and columns with the cost of a subfamily. For a subfamily $\mathcal{H}$, let $p(\mathcal{H})$ denote the total profit of elements uniquely covered by $\mathcal{H}$. For $\mathcal{C}'_g \subseteq \mathcal{C}_g$ and $1 \leq i \leq B$, let $\mathcal{W}_i[\mathcal{C}'_g]$ denote the set of all subfamilies of budget at most i which has a good configuration w.r.t. h and $\mathcal{C}'_g$.

For $\mathcal{C}'_g \subseteq \mathcal{C}_g$ and $1 \leq i \leq B$, define $W[\mathcal{C}'_g][i]$ as follows:

$$W[\mathcal{C}'_g][i] = \begin{cases} \max_{\mathcal{H} \in \mathcal{W}_i[\mathcal{C}'_g]}\{p(\mathcal{H})\}, & \text{if } \mathcal{W}_i[\mathcal{C}'_g] \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases}$$

The entry corresponding to $W[\emptyset][i]$ is set to 0 for all $1 \leq i \leq B$ as a convention. We fill this array in increasing order of sizes of the subsets of $\mathcal{C}_g$. For $S \in \mathcal{F}$, define $g(S) = h(S) \cap \mathcal{C}_g$ to be the set of good colors assigned to elements in S and $p(S)$ to be the total profit of elements in S assigned colors from $g(S)$. Let $\mathcal{T}$ be the subfamily containing all sets $S \in \mathcal{F}$ such that $g(S) \subseteq \mathcal{C}'_g$ and for each $c \in g(S)$ there exists exactly one element $e \in S$ with $h(e) = c$. Then

$$W[\mathcal{C}'_g][i] = \max_{S \in \mathcal{T}} \left\{ p(S) + W[\mathcal{C}'_g \setminus g(S)][i - c(S)] \right\}$$

The correctness of the algorithm is immediate. Clearly, if $W[\mathcal{C}_g][B] \geq k$, then there exists a subfamily with total cost at most B which uniquely covers elements with total profit at least k. Such a family can be found by simply storing the witness set for every entry in the table and backtracking. The time taken by the algorithm is $O(2^k Bmk + |\mathcal{C}|) = O(2^k Bmk)$, where $m = |\mathcal{F}|$.

Lemma 5.14. *Let $(\mathcal{U}, \mathcal{F}, B, k, c, p)$ be an instance of* Budgeted Unique Coverage *with integral costs and profits, and $h\colon \mathcal{U} \to \mathcal{C}$ be a coloring function. Then one can find a subfamily $\mathcal{H}$ of total cost at most B which has a good configuration w.r.t. coloring h and $\mathcal{C}_g$, if there exists one, in time $O(2^k Bmk)$.*

Note that Lemma 5.14 holds even when profits are from $\mathbb{Q}^{\geq 1}$. When the costs are from $\mathbb{Q}^{\geq 1}$ and profits are positive integers then we can modify the dynamic programming algorithm as follows. Define an array W of size $2^k \times k$ and identify its rows with subsets of $\mathcal{C}_g$ and columns with the profit of a subfamily. For $\mathcal{C}'_g \subseteq \mathcal{C}_g$ and $1 \leq i \leq k$, define $\mathcal{W}_i[\mathcal{C}'_g]$ and $W[\mathcal{C}'_g][i]$ as follows: $\mathcal{W}_i[\mathcal{C}'_g]$ denotes the set of all subfamilies with profit at least i and which have a good configuration w.r.t. the given coloring function h and $\mathcal{C}'_g$; and

$$W[\mathcal{C}'_g][i] = \begin{cases} \min_{\mathcal{H} \in \mathcal{W}_i[\mathcal{C}'_g]}\{c(\mathcal{H})\}, & \text{if } \mathcal{W}_i[\mathcal{C}'_g] \neq \emptyset; \\ \infty, & \text{otherwise.} \end{cases}$$

The entry $W[\emptyset][i]$ is set to 0 for all $1 \leq i \leq k$. Any entry of the form $W[\mathcal{C}'_g][i]$ where $\mathcal{C}'_g \subseteq \mathcal{C}_g$ and $i \leq 1$ is identified with the entry $W[\mathcal{C}'_g][1]$. Given $\mathcal{C}'_g \subseteq \mathcal{C}_g$, let $\mathcal{T}$ be the subfamily containing all sets $S \in \mathcal{F}$ such that $g(S) := h(S) \cap \mathcal{C}_g \subseteq \mathcal{C}'_g$ and for each $c \in g(S)$ there exists exactly one $e \in S$ with $h(e) = c$. Then

$$W[\mathcal{C}'_g][i] = \min_{S \in \mathcal{T}}\{c(S) + W[\mathcal{C}'_g \setminus g(S)][i - p(S)]\}.$$

If $W[\mathcal{C}_g][k] \leq B$ then there exists a subfamily with total cost at most B that uniquely covers elements with total profit at least k. The time taken by the algorithm is $O(2^k \cdot mk^2)$. We thus obtain:

Theorem 5.14. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of the* Budgeted Unique Coverage *problem with either integral costs and rational profits ≥ 1 or with rational costs ≥ 1 and integral profits. Then one can find a subfamily $\mathcal{F}'$ of total cost at most B that uniquely covers elements with total profit at least k, if there exists one, in time $O(\min\{2^{O(k \log B)}, 2^{O(k \log k + k \log\log B)}\} \cdot Bmk^2 \log^2 n)$.*

5.6 Algorithms for Special Cases

We now consider deterministic algorithms for two special cases of Budgeted Unique Coverage: Unique Coverage and Budgeted Max Cut.

5.6.1 Unique Coverage

An instance $(\mathcal{U}, \mathcal{F}, k)$ of Unique Coverage can be viewed as an instance of Budgeted Unique Coverage where the costs and profits are all one and the budget $B = k$ as we do not need more than k sets to cover k elements uniquely. Using Theorem 5.12, we immediately obtain an algorithm with running time

$$O(2^{O(k \log k)} \cdot |\mathcal{F}| \cdot k^2 \log^2 n).$$

In this subsection we present an algorithm for Unique Coverage that runs in deterministic time

$$O(2^{O(k \log\log k)} \cdot |\mathcal{F}| \cdot k + |\mathcal{F}|^2).$$

We first need some lower bounds on the number of elements that can be uniquely covered in any instance of Unique Coverage.

Define the *frequency* f_u of an element $u \in \mathcal{U}$ to be the number of sets in the family $\mathcal{F}$ that contain u. Let γ denote the maximum frequency, that is, $\gamma = \max_{u \in \mathcal{U}}\{f_u\}$.

Lemma 5.15. *Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of* Unique Coverage*, one can in polynomial time obtain a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that $\mathcal{F}'$ covers at least $|\mathcal{U}|/(4e \log \gamma)$ elements uniquely, where $\gamma = \max_{u \in \mathcal{U}}\{f_u\}$.*

Proof. The proof of lemma is similar to the Lemma 5.8 but we provide a proof here for the sake of completeness.

Partition the elements of $\mathcal{U}$ into sets $C_0, C_1 \ldots C_{r-1}$, such that $u \in C_i$ if $2^i \leq f_u \leq 2^{i+1} - 1$. Note that i ranges from 0 to $\log(\gamma + 1) - 1$, since the frequency of any element in C_{r-1} is at most $2^r - 1$, which in turn is at

most γ. So the total number of sets that the universe gets partitioned into is $\log(\gamma+1)$.

Clearly, there exists j such that $|C_j| \geq n/\log(\gamma+1)$. Fix j to be the index for which $|C_j| \geq n/\log(\gamma+1)$. Let $u \in C_j$ and note that $2^j \leq f_u \leq 2^{j+1}-1$. Construct a subfamily $\mathcal{F}'$ from $\mathcal{F}$ by going through each set in $\mathcal{F}$ and including it in $\mathcal{F}'$ with probability $1/2^{j+1}$. Denote by l_u the probability that u is covered uniquely by $\mathcal{F}'$. Then

$$\begin{aligned} l_u &= \left(\tfrac{f_u}{2^{j+1}}\right)\left(1-\tfrac{1}{2^{j+1}}\right)^{(f_u-1)} \\ &\geq \tfrac{1}{2}\left(1-\tfrac{1}{2^{j+1}}\right)^{(f_u-1)} && \text{(since } f_u \geq 2^j\text{).} \\ &\geq \tfrac{1}{2}\left(1-\tfrac{1}{2^{j+1}}\right)^{(2^{j+1}-2)} && \text{(since } f_u \leq 2^{j+1}-1\text{).} \\ &\geq \tfrac{1}{2}\left(1-\tfrac{1}{2^{j+1}}\right)^{(2^{j+1})} \geq \tfrac{1}{4e}. && \text{(by Lemma 5.7).} \end{aligned}$$

Now, let X_u be an indicator random variable which takes value 1 if u is covered uniquely by the subfamily $\mathcal{F}'$, and 0 otherwise. Also, define $X = \sum_u X_u$.

$$E(X) = \sum_{u \in \mathcal{U}} E(X_u) \geq \sum_{u \in C_j} E(X_u) = \sum_{u \in C_j} l_u \geq \frac{n}{4e \log \gamma}.$$

This immediately implies the existence of a subfamily that covers $n/4e\log\gamma$ elements (or more) uniquely. We can derandomize the above algorithm using the method of conditional probabilities to obtain the desired subfamily. □

Lemma 5.16. *Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of* Unique Coverage, *one can in polynomial time obtain a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that $\mathcal{F}'$ covers at least $|\mathcal{U}|/(8e \log M)$ elements uniquely, where $M = \max_{S \in \mathcal{F}}\{|S|\}$.*

Proof. We begin by constructing a subfamily $\mathcal{F}'$ from $\mathcal{F}$ that is *minimal* in the sense that every set in $\mathcal{F}'$ covers at least one element in $\mathcal{U}$ uniquely. Such a subfamily is easily obtained, by going over every set in the family and checking if it has at least one element which is not contained in any other set. Let m' denote the size of the subfamily $\mathcal{F}'$. Note that $|\bigcup_{S \in \mathcal{F}'} S| = n$. For the proof of the lemma we distinguish two cases based on m':

Case 1: ($m' \geq n/2$) As the subfamily is minimal, by construction, we are immediately able to cover at least $n/2$ elements uniquely. Thus $\mathcal{F}'$ itself satisfies the claim of the lemma.

Case 2: ($m' < n/2$) In this case, we first claim that $|\{u \in \mathcal{F}' : f_u < M\}| \geq n/2$. If not, then there would be more than $n/2$ elements whose frequency is at least M, which implies that $\sum_{S \in \mathcal{F}'} |S| > Mn/2$. On the other hand, $\sum_{S \in \mathcal{F}'} |S|$ is clearly at most $M(n/2-1)$ (because there are strictly less than $n/2$ sets in the family and the size of any set in the family is bounded

by M). The claim implies that there exists a set of at least $n/2$ elements whose frequency is less than M. Denote this set of elements by $\mathcal{V}$. Consider the family $\mathcal{F}''$ obtained from $\mathcal{F}'$ as follows: $\mathcal{F}'' = \{S \cap \mathcal{V} \mid S \in \mathcal{F}'\}$. Applying Lemma 5.15 to the instance $(\mathcal{V}, \mathcal{F}'')$, we obtain a subfamily $\mathcal{T}$ of $\mathcal{F}''$ that covers at least $n/(8e \log M)$ elements uniquely. The corresponding subfamily of $\mathcal{F}'$ will clearly cover the same set of elements uniquely in $\mathcal{U}$. This completes the proof of the lemma. □

Using these lower bounds on the number of elements that are uniquely covered, we can upper bound the size of a YES-instance of the UNIQUE COVERAGE problem as a function of the parameter k. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of UNIQUE COVERAGE reduced w.r.t. Rules R1 and R2 described in Section 5.4. Then $|S| \leq k-1$ for all $S \in \mathcal{F}$ and M is bounded above by $k-1$. If $k \leq n/8e \log(k-1)$, then there exists a subfamily that covers k elements uniquely. If not, we have $k > n/8e \log k$, which implies that $n < 8ek \log k$.

Lemma 5.17. *Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of* UNIQUE COVERAGE. *Then, in polynomial time, one can either find a subfamily covering at least k elements uniquely, or an equivalent instance where the size of the universe is $O(k \log k)$.*

Note that the above lemma shows that UNIQUE COVERAGE admits a kernel with $k^{O(k)}$ sets which is what was shown in Corollary 5.2.

An improved algorithm for UNIQUE COVERAGE first applies Lemma 5.17 and obtains an instance of UNIQUE COVERAGE, $(\mathcal{U}, \mathcal{F}, k)$, where $n = |\mathcal{U}| \leq O(k \log k)$. Now we examine all k-sized subsets X of the universe $\mathcal{U}$ and check whether there exists a subfamily that covers it uniquely. Let $X = \{u_{i_1}, u_{i_2}, \ldots, u_{i_k}\}$, and let h be a function that maps X injectively to $[k]$ and each element in $\mathcal{U} \setminus X$ to the color $k+1$. Applying Lemma 5.10 to the instance $(\mathcal{U}, \mathcal{F}, B = k, k)$, with the coloring function h described above gives us an algorithm to find the desired $\mathcal{F}'$ in time $O(2^k k^2 m)$. Note that a factor of k can be avoided by directly applying dynamic programming over subsets of X. The size of $\mathcal{U}$ is upper bounded by $8ek \log k$ and hence the total number of subsets that need to be examined is at most $\binom{8ek \log k}{k}$, which is bounded above by $(8e \log k)^k \leq 2^{4.5k + k \log\log k}$. Combining this with the above discussion results in:

Theorem 5.15. *Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of* UNIQUE COVERAGE. *Then one can check whether there exists a subfamily covering at least k elements of $\mathcal{U}$ uniquely in time $O(2^{5.5k + k \log\log k} \cdot mk + m^2)$.*

5.6.2 Budgeted Max Cut

An instance of BUDGETED MAX CUT consists of an undirected graph $G = (V, E)$ on n vertices and m edges; a cost function $c\colon V \to \mathbb{Z}^+$; a profit function $p\colon E \to \mathbb{Z}^+$; and positive integers k and B. The question is whether

there exists a cut $(T, V \setminus T)$, $\emptyset \neq T \neq V$, such that the total cost of the vertices in T is at most B and the total profit of the edges crossing the cut is at least k. This problem can be modelled as an instance Budgeted Unique Coverage by taking $\mathcal{U} = E$ and $\mathcal{F} = \{S_v \colon v \in V\}$, where $S_v = \{e \in E \colon e \text{ is incident on } v\}$.

The algorithm we describe has running time $O(2^{O(k)} \cdot Bmk \cdot \log^2 n)$. Given $S \subseteq V$, we let $c(S)$ denote the total cost of the elements of S. If $(S, V \setminus S)$ is a cut in a graph G, then $p(S, V \setminus S)$ is the total profit of edges across the cut. Define the profit $\hat{p}(v)$ of a vertex v to be the sum of the profits of all the edges incident on it. We assume that $\hat{p}(v) \leq k - 1$ for all $v \in V$, for otherwise the given instance is a yes-instance trivially.

Lemma 5.18. *If (G, B, k, c, p) is a* yes*-instance of* Budgeted Max Cut *then there exists a cut $(S, S \setminus V)$ such that $c(S) \leq B$, $p(S, V \setminus S) \geq k$, and $|\bigcup_{v \in S} S_v| \leq 4k$.*

Proof. Since we are given a yes-instance of the problem, there exists a cut (T, T') such that $c(T) \leq B$ and $p(T, T') \geq k$. Call a vertex v of T *redundant* if $p(T \setminus v, T' \cup v) \geq k$. Starting with the cut (T, T'), transfer redundant vertices from T to T' and obtain a cut (S, S') such that $S \subset T$ and S does not contain any redundant vertices. Observe that $c(S) \leq B$ and $p(S, S') \geq k$. For any $v \in S$, $\hat{p}(v) \leq k - 1$ and $p(S \setminus v, S' \cup v) \leq k - 1$. Therefore $p(S, S') \leq 2k$. For $v \in S$, partition S_v as $I_v \uplus C_v$, where I_v is the set of edges incident on v that lie entirely in S and C_v are the edges that lie across the cut (S, S'). Clearly $p(I_v) \leq p(C_v)$, for otherwise, $p(S \setminus v, S' \cup v) > p(S, S')$, a contradiction to the fact that S has no redundant vertices. Therefore $\sum_{v \in S} p(I_v) \leq \sum_{v \in S} p(C_v) \leq 2k$. This yields $\sum_{v \in S} \hat{p}(v) \leq 4k$. Since the profits are at least one, we have $|\bigcup_{v \in S} S_v| \leq 4k$. □

We now use the deterministic algorithm outlined before Theorem 5.12 with $t = s = 4k$ and a $4k$-perfect hash family by Chen et al. [32]. The running time then works out to $O(6.4^{4k} \log^2 n \cdot \binom{4k}{k} \cdot 2^k Bmk)$ which simplifies to $O(2^{13.8k} \cdot Bmk \cdot \log^2 n)$.

Theorem 5.16. *Let (G, B, k, c, p) be an instance of* Budgeted Max Cut*. Then one can find a cut (S, S') such that $c(S) \leq B$ and $p(S, S') \geq k$, if there exists one, in time $O(2^{13.8k} \cdot Bmk \cdot \log^2 n)$.*

5.7 Conclusions

In this chapter we studied the parameterized complexity of the Unique Coverage problem. We considered several plausible parameterizations of the problem and showed that except for the standard parameterized version (and a generalization of it), all other versions are unlikely to be fixed-parameter intractable. We also described problem kernels for several special

Unique Coverage (*Parameter: k*)	*Kernel Size*	*Sect.*
Each element occurs in at most b sets	$(k-1)b$	5.4
Intersection size bounded by c	k^{c+1}	5.4.1
General case	4^k	5.4.2
Each set of size at most b	2^{b+k}	5.4.2
Budgeted Unique Coverage	*Complexity*	*Sect.*
Arbitrary weights (parameters B and k)	Not FPT (unless P = NP)	5.5.1
Integer weights (parameter B)	W[1]-hard	5.5.1
Integer weights (parameters B and k)	FPT	5.5.2

Table 5.1: Main results in this chapter.

cases of the standard parameterized version of Unique Coverage and showed that the general version admits a kernel with at most 4^k sets. We noted that this is essentially the best possible kernel due to a lower bound result by Dom et al. [46].

We gave fixed-parameter tractable algorithms for Budgeted Unique Coverage and several of its variants. Our algorithms were based on an application of the well-known method of color-coding in an interesting way. Our randomized algorithms have good running times but the deterministic algorithms make use of either perfect hash families or (k, s)-hash families and this introduces large constants in the running times, a common enough phenomenon when derandomizing randomized algorithms using such function families [6]. The main results of this chapter are summarized in Table 5.1.

Our use of (k, s)-hash families to derandomize algorithms is perhaps the first application of these hash families outside the domain of coding theory and it suggests that this artificial-looking class of hash families may be important and may have other uses as well. It will be interesting to give explicit constructions of (k, s)-hash families of size promised by Lemma 5.12 and explore other applications of our generalization of the color-coding technique.

Chapter 6

The Induced Subgraph Problem in Directed Graphs

In this chapter and the next, we consider the parameterized complexity of NP-optimization problems in directed graphs. There have been relatively few results on parameterized problems on directed graphs since, in general, many problems that can be formulated for both directed and undirected graphs are significantly more difficult for directed graphs [78]. For instance, the Feedback Edge Set problem is polynomial-time solvable in undirected graphs but NP-complete in directed graphs [64]. From the parameterized complexity point of view, the Undirected Feedback Vertex Set problem is known to be fixed-parameter tractable but the Directed Feedback Vertex (Edge) Set was open for a number of years until it was shown to be in FPT by Chen et al. [31]. The results in this chapter and the next add to the growing literature on parameterized complexity results on directed graphs.

The problem that we consider in this chapter actually represents a class of problems that is loosely termed as the Induced Subgraph problem and is defined as follows: Given a graph G and a nonnegative integer k, does G have a vertex induced subgraph of size k satisfying some prespecified property? Lewis and Yannakakis [93] proved that this problem is NP-complete when the property is nontrivial and hereditary. Khot and Raman [86] studied the parameterized complexity of this problem in undirected graphs and completely characterized properties for which the problem is in FPT and for which the problem is W[1]-complete. We extend their result for hereditary properties on directed graphs. As a corollary of our results, we show, for example, that the problem of deciding whether an input digraph D has a transitive induced subdigraph of size k is fixed parameter tractable while the problem of deciding whether D has a planar induced subdigraph of size k is W[1]-complete.

This chapter is organized as follows. In Section 6.1, we define the problem formally, briefly survey some previous work and define the digraph classes

that appear in this chapter. In Section 6.2, we give a complete specification of when the INDUCED SUBGRAPH problem is fixed-parameter tractable and when it is not, for hereditary properties on general directed graphs. In Section 6.3, we consider the problem for hereditary properties on oriented graphs. In Section 6.4, we characterize those hereditary properties for which the INDUCED SUBGRAPH problem is hard on general digraphs but in FPT on oriented graphs. We end with some concluding remarks in Section 6.5.

6.1 Problem Definition and Previous Work

A *graph property* $\mathcal{P}$ is an isomorphism-closed set of graphs. A graph property $\mathcal{P}$ is *nontrivial* if there exists an infinite family of graphs satisfying $\mathcal{P}$ and an infinite family not satisfying $\mathcal{P}$. A graph property $\mathcal{P}$ is *hereditary* if $G \in \mathcal{P}$ implies that every induced subgraph of G is also in $\mathcal{P}$ (see [93]). Examples of hereditary properties (for undirected graphs) include the class of planar, outerplanar, bipartite, interval, comparability, acyclic, bounded-degree, chordal, complete, independent set and line invertible graphs [93]. Similarly for digraphs, the following graph classes are hereditary: acyclic, transitive, symmetric, anti-symmetric, line-digraph, maximum outdegree r, maximum indegree r, without cycles of length l, without cycles of length $\leq l$ [93].

A property $\mathcal{P}$ has a *forbidden set characterization* if there exists a set $\mathcal{F}$ of graphs such that G has property $\mathcal{P}$ if and only if no element of $\mathcal{F}$ is an induced subgraph of G. The set $\mathcal{F}$ is called the *forbidden set* of $\mathcal{P}$. It is well-known that a property $\mathcal{P}$ is hereditary if and only if it has a forbidden set characterization [25]. For if a property $\mathcal{P}$ has a forbidden set characterization, it is clearly hereditary. Conversely suppose $\mathcal{P}$ is hereditary and consider the set $\mathcal{S}$ of graphs *not* in $\mathcal{P}$. The induced subgraph relation defines a partial order among the elements of $\mathcal{S}$ and the minimal elements of this partial order form the forbidden set of $\mathcal{P}$.

For a property $\mathcal{P}$ on (directed) graphs, the INDUCED SUBGRAPH problem is defined as follows: Given a (directed) graph G find a vertex subset of maximum size that induces a subgraph with property $\mathcal{P}$. Lewis and Yannakakis [93] proved this problem to be NP-hard when the property $\mathcal{P}$ is nontrivial and hereditary. If, in addition, the given property can be tested in polynomial time, their results show that the INDUCED SUBGRAPH problem is NP-complete. The parameterized version of this problem for a given property $\mathcal{P}$ is defined as follows.

$P(G, k, \mathcal{P})$

Input:	A graph $G = (V, E)$ and a nonnegative integer $k \leq \|V\|$.
Parameter:	The integer k.
Question:	Does G have an induced subgraph on at least k vertices with property $\mathcal{P}$?

Call the directed graphs version of this problem $P(D, k, \mathcal{P})$.

Khot and Raman [86] resolved the problem $P(G, k, \mathcal{P})$ when $\mathcal{P}$ is a nontrivial hereditary property on undirected graphs. They show that if the property $\mathcal{P}$ either contains all trivial graphs and all cliques or excludes a trivial graph and a clique then the problem $P(G, k, \mathcal{P})$ is fixed-parameter tractable and W[1]-complete otherwise. The proof techniques employed by them make heavy use of Ramsey theory. In particular, they make use of the fact that any "sufficiently large" undirected graph either contains a trivial graph or a clique.

In this chapter we consider the problem $P(D, k, \mathcal{P})$ when $\mathcal{P}$ is a nontrivial hereditary property on directed graphs. We give a complete specification of when the problem $P(D, k, \mathcal{P})$ is fixed-parameter tractable and when it is not. At this point, we define a few graph classes that we will encounter in this chapter (see [12] for details). A graph is *trivial* if it has no edges (or arcs). A graph without cycles is *acyclic*. A directed graph $D = (V, A)$ is *complete symmetric* if for all distinct $u, v \in V$, we have $(u, v) \in A$ and $(v, u) \in A$. A directed graph is a *tournament* if for all vertex pairs, there exists exactly one arc between them. A directed graph with at most one arc between each vertex-pair is called an *oriented* graph. That is, an oriented graph is one which does not have cycles of length two. Oriented graphs are also called *antisymmetric*. A digraph $D = (V, A)$ is *symmetric* if $(u, v) \in A$ implies $(v, u) \in A$.

A set K of vertices in a digraph $D = (V, A)$ is a *kernel* if K is independent and the closed neighbourhood $N[K]$ of K is V itself. A digraph is *kernel-perfect* if every induced subdigraph has a kernel. Clearly all trivial and complete symmetric digraphs are kernel-perfect, as any maximal independent set is a kernel. What is interesting is that any acyclic digraph is kernel perfect too [12]. A digraph is chordal if its underlying undirected graph is chordal, that is, if it has no induced cycles of length greater than three. A digraph $D = (V, A)$ is *transitive* if for distinct vertices $u, v, w \in V$, $(u, v) \in A$ and $(v, w) \in A$ imply that $(u, w) \in A$. A digraph is *quasi-transitive* if for all distinct vertices u, v, w, if $(u, v) \in A$ and $(v, w) \in A$ then there is at least one arc between u and w.

6.2 General Directed Graphs

We begin with by examining a specialization of Ramsey's theorem applicable to directed graphs.

Fact 6.1 ([69])**.** *Suppose that the 2-element subsets of every set S with n elements are partitioned into m disjoint families $\mathscr{F}_1, \ldots, \mathscr{F}_m$. Let $p_1, \ldots, p_m$ be positive integers with $p_i \geq 2$, $1 \leq i \leq m$. Then there is a number $r(p_1, \ldots, p_m)$, such that for every set S with $n \geq r(p_1, \ldots, p_m)$ elements*

there exists an i, $1 \leq i \leq m$, and a subset A_i of S with p_i elements all of whose 2-subsets are in the family $\mathscr{F}_i$.

If $D = (V, A)$ is a digraph with $V = \{u_1, \ldots, u_n\}$, partition the 2-subsets of V into four classes, as follows:

$$\begin{aligned}
\mathscr{F}_1 &= \{\{u_i, u_j : (u_i, u_j), (u_j, u_i) \notin A\}\} \\
\mathscr{F}_2 &= \{\{u_i, u_j : (u_i, u_j), (u_j, u_i) \in A\}\} \\
\mathscr{F}_3 &= \{\{u_i, u_j : (u_i, u_j) \in A, (u_j, u_i) \notin A, i < j\}\} \\
\mathscr{F}_4 &= \{\{u_i, u_j : (u_i, u_j) \notin A, (u_j, u_i) \in A, i < j\}\}
\end{aligned}$$

From Ramsey's theorem we have

Corollary 6.1. *Let p_1, p_2, p_3 be positive natural numbers ≥ 2. Then there exists a positive number $r(p_1, p_2, p_3)$ such that any directed graph D on at least $r(p_1, p_2, p_3)$ vertices contains either a trivial graph of size p_1, or a complete symmetric digraph of size p_2, or an acyclic tournament of size p_3.*

Thus if $\mathcal{P}$ is a nontrivial hereditary property on digraphs, then it must contain either all trivial graphs, all complete symmetric digraphs and all acyclic tournaments or exactly two of these graph types of all sizes or exactly one of these graph types of all sizes.

Theorem 6.1. *If $\mathcal{P}$ is a hereditary property on digraphs that either contains all trivial graphs, all complete symmetric digraphs and all acyclic tournaments or excludes a graph of each of these three types, then the problem $P(D, k, \mathcal{P})$ is fixed-parameter tractable.*

Proof. Suppose $\mathcal{P}$ excludes a trivial graph of size c_1, a complete symmetric digraph of size c_2 and an acyclic tournament of size c_3. Then $\mathcal{P}$ cannot contain any digraph D such that $|V(D)| \geq r(c_1, c_2, c_3)$ and is therefore finite. The problem $P(D, k, \mathcal{P})$ can then be decided in constant time.

Therefore assume that $\mathcal{P}$ contains all trivial graphs, all complete symmetric digraphs and all acyclic tournaments. If $|V(D)| \geq r(k, k, k)$[1] then, by Ramsey's theorem, the digraph D has either a trivial graph, or a complete symmetric digraph or an acyclic tournament of size k as an induced subgraph. Thus the given instance is a YES-instance. Otherwise, $|V(D)| < r(k, k, k)$ and we check all subsets $S \subseteq V(D)$ of size k to see whether $D[S]$ has property $\mathcal{P}$. This takes time $\binom{r(k,k,k)}{k} \cdot f(k)$, where $f(k)$ is the time taken to decide whether a digraph on k vertices has property $\mathcal{P}$. This proves that the problem $P(D, k, \mathcal{P})$ is fixed-parameter tractable. □

[1] We do not need to know the number $r(k, k, k)$ exactly. An upper bound on $r(k, k, k)$ will serve our purpose. A crude upper-bound for the Ramsey number $r(\underbrace{k, \ldots, k}_{s \text{ times}})$ is s^{sk}.

Corollary 6.2. *Given any directed graph D and an integer k, it is fixed-parameter tractable to decide whether D has an induced subdigraph on k vertices that is (1) a kernel perfect digraph, (2) a chordal digraph, (3) a transitive digraph, or (4) a quasi-transitive digraph.*

6.2.1 W[1]-Completeness Results

We show that if the property $\mathcal{P}$ contains exactly two of the graph types of all sizes or exactly one of the graph types of all sizes then the problem $P(D, k, \mathcal{P})$ is W[1]-complete. To do this, we first show that the problem $P(D, k, \mathcal{P})$ is in W[1] for any nontrivial decidable hereditary property $\mathcal{P}$. We next show that the problem is W[1]-hard by exhibiting a parametric reduction from a W[1]-hard problem.

Lemma 6.1. *Let $\mathcal{P}$ be a nontrivial decidable hereditary property on digraphs. Then the problem $P(D, k, \mathcal{P})$ is in* W[1].

Proof. We reduce $P(D, k, \mathcal{P})$ to the Short Turing Machine Acceptance problem (defined below) which is complete for the class W[1] [47] (also see Chapter 1).

Input:	A nondeterministic Turing machine M, a string x and non-negative integer k
Parameter:	The integer k.
Question:	Does M have a computation path accepting x in at most k steps?

Let $(D = (V, E), k)$ be an instance of $P(D, k, \mathcal{P})$, with $|V| = n$. We will show that we can construct an instance (M_D, x, k') of Short Turing Machine Acceptance in time $O(f(k) \cdot n^{O(1)})$ such that D has an induced subgraph of size k satisfying property $\mathcal{P}$ if and only if M_D accepts x within k' steps, where k' depends only on k.

First note that since we assumed $\mathcal{P}$ to be decidable, there exists a deterministic Turing machine (DTM) M' that takes a digraph D as input and in time $t(|V(D)|)$ decides whether D satisfies $\mathcal{P}$. The input alphabet of M_D consists of the $n + 1$ symbols $1, 2, 3, \ldots, n, \sharp$. The NTM M_D performs the following steps.

1. M_D nondeterministically writes a sequence of k numbers on its tape out of its tape alphabet $\{1, 2, \ldots, n\}$.

2. It then verifies whether the k numbers it has picked are distinct.

3. It then constructs the subgraph D' of D represented by these k vertices.

4. M_D passes control to M' which then verifies whether D' satisfies $\mathcal{P}$. If yes, M_D accepts.

The time taken in Steps 1, 2 and 4 are, respectively, $O(k)$, $O(k^2)$ and $t(k)$. Assuming that the graph D is hardwired in M_D as an adjacency matrix, Step 3 takes time $O(k^2)$.

It is easy to see that (D, k) is a YES-instance of the problem $P(D, k, \mathcal{P})$ if and only if the NTM M_D accepts the empty string in $k' = O(k+k^2+t(k))$ steps. □

To prove W[1]-hardness, we consider the following four cases: the property $\mathcal{P}$ contains

1. all complete symmetric digraphs but not all trivial graphs;

2. all trivial graphs but not all complete symmetric digraphs;

3. all acyclic tournaments but not all trivial graphs;

4. all trivial graphs but not all acyclic tournaments.

Note that cases 1 through 4, though not mutually exclusive, are exhaustive.

We first show that $P(D, k, \mathcal{P})$ is W[1]-hard in cases 1 and 2.

Theorem 6.2. *Let $\mathcal{P}$ be a hereditary property on digraphs that contains all complete symmetric digraphs but not all trivial graphs or vice versa. Then $P(D, k, \mathcal{P})$ is* W[1]*-complete.*

Proof. Membership in W[1] was shown in Lemma 6.1. We therefore need only establish W[1]-hardness.

Let $\mathcal{P}$ be a property on digraphs. Define $\mathcal{P}_1$ as follows. An undirected graph $G \in \mathcal{P}_1$ if and only if the directed graph D obtained from G by replacing every edge $\{u, v\} \in E(G)$ by the arcs (u, v) and (v, u) is in $\mathcal{P}$. Note that G contains a clique of size k if and only if D contains a complete symmetric digraph of size k and G contains an independent set of size k if and only if D contains an independent set of size k. Also note that

1. $\mathcal{P}_1$ is nontrivial and hereditary if and only if $\mathcal{P}$ is nontrivial and hereditary,

2. $\mathcal{P}_1$ contains all cliques but not all trivial graphs if and only if $\mathcal{P}$ contains all complete symmetric digraphs but not all trivial graphs, and

3. $\mathcal{P}_1$ contains all trivial graphs but not all cliques if and only if $\mathcal{P}$ contains all trivial graphs but not all complete symmetric digraphs.

By Khot and Raman [86], the problem $P_1(G, k, \mathcal{P}_1)$ is W[1]-hard when $\mathcal{P}_1$ contains all cliques but not all trivial graphs or vice versa.

We now exhibit a parametric reduction from $P_1(G, k, \mathcal{P}_1)$ to $P(D, k, \mathcal{P})$. Let (G, k) be an instance of P_1. Construct a directed graph D as follows: $V(D) = V(G)$ and for all $u, v \in V(G)$, if $\{u, v\} \in E(G)$ add the arcs (u, v) and (v, u) in $A(D)$. The directed graph D has no other arcs. From the manner in which property $\mathcal{P}_1$ was defined, it is clear that G has an induced subgraph on k vertices satisfying $\mathcal{P}_1$ if and only if D has an induced subdigraph on k vertices satisfying $\mathcal{P}$. This completes the proof. □

We next show that $P(D, k, \mathcal{P})$ is W[1]-hard in cases 3 and 4.

Theorem 6.3. *Let $\mathcal{P}$ be a hereditary property on digraphs that contains all acyclic tournaments but not all trivial graphs or vice versa. Then the problem $P(D, k, \mathcal{P})$ is* W[1]*-complete.*

Proof. As before, we show only W[1]-hardness. Let $\mathcal{P}$ be a property on digraphs. Define $\mathcal{P}_1$ to be a set of undirected graphs with the following property: An undirected graph $G \in \mathcal{P}_1$ if and only if the directed graph $D \in \mathcal{P}$, where $V(D) = V(G)$ with an ordering on the vertices and

$$A(D) = \{(u, v) \colon u < v \text{ and } \{u, v\} \in E(G)\}.$$

Clearly G has an independent set of size k if and only if D has an independent set of size k and G has a clique of size k if and only if D has an acyclic tournament of size k. Also

1. $\mathcal{P}_1$ is nontrivial and hereditary if and only if $\mathcal{P}$ is nontrivial and hereditary,

2. $\mathcal{P}_1$ contains all trivial graphs but not all cliques if and only if $\mathcal{P}$ contains all trivial graphs but not all acyclic tournaments, and

3. $\mathcal{P}_1$ contains all cliques but not all trivial graphs if and only if $\mathcal{P}$ contains all acyclic tournaments but not all trivial graphs.

The problem $P_1(G, k, \mathcal{P}_1)$ is W[1]-hard by [86] when $\mathcal{P}_1$ contains all trivial graphs but not all cliques or vice versa.

We now exhibit a parametric reduction from the problem $P_1(G, k, \mathcal{P}_1)$ to the problem $P(D, k, \mathcal{P})$. Given an instance (G, k) of $P_1(G, k, \mathcal{P}_1)$, let D be the directed graph obtained by orienting the edges of G from lower ordered vertices to higher ordered vertices. From the manner in which we constructed $\mathcal{P}_1$, it is easy to see that G has an induced subgraph on k vertices satisfying $\mathcal{P}_1$ if and only if D has an induced subdigraph on k vertices satisfying $\mathcal{P}$. This proves the theorem. □

We now look at some applications. The set of symmetric digraphs contains all trivial graphs and all complete symmetric digraphs but no acyclic tournament. The following hereditary properties contain all trivial graphs and acyclic tournaments but not all complete symmetric digraphs: (1) acyclic digraphs, (2) oriented digraphs, (3) digraphs without dicycles of length l and (4) digraphs without dicycles of length $\leq l$. Hence the following corollary is immediate from Theorems 6.2 and 6.3.

Corollary 6.3. *Given a digraph D and a positive integer k, it is* W[1]*-complete to decide whether D has an induced subdigraph of size k that is (1) a symmetric digraph, (2) acyclic, (3) an oriented digraph, (4) without dicycles of length l, or (5) without dicycles of length $\leq l$.*

The following digraph properties contain all trivial graphs but not all complete symmetric digraphs and acyclic tournaments: (1) with maximum indegree r, (2) with maximum outdegree r, (3) bipartite, (4) colorable with c colors, for some constant $c \geq 1$, (5) planar, (6) a line digraph. Hence the following corollary is immediate from Theorem 6.3.

Corollary 6.4. *Given a digraph D and a positive integer k, it is* W[1]*-complete to decide whether D has an induced subdigraph of size k that is (1) of maximum indegree r, (2) of maximum outdegree r, (3) bipartite, (4) colorable with c colors, for some constant $c \geq 1$, (5) planar, or (6) a line digraph.*

6.3 Oriented Graphs

Though Corollary 6.3 says that finding an acyclic subdigraph of size at least k is hard in general digraphs, Raman and Saurabh [114] have shown that this problem is in FPT in oriented graphs. In this section, we look at the general Induced Subgraph problem in oriented graphs.

Recall that an oriented graph is a directed graph in which every pair of vertices has at most one arc between them. Thus oriented graphs are precisely those digraphs with no 2-cycle. For oriented graphs, Ramsey's theorem says: For positive integers p and q there exists an integer $r(p, q) \in \mathbb{N}$ such that any oriented graph on at least $r(p, q)$ vertices either has a trivial graph of size p or an acyclic tournament of size q.

Any nontrivial hereditary property $\mathcal{P}$ on oriented graphs can therefore be classified into one of the three types: (1) $\mathcal{P}$ contains all trivial graphs and all acyclic tournaments; (2) $\mathcal{P}$ contains all trivial graphs but not all acyclic tournaments; (3) $\mathcal{P}$ contains all acyclic tournaments but not all trivial graphs. As one might suspect, the problem $P(D, k, \mathcal{P})$ is fixed-parameter tractable for Case (1) and W[1]-complete for Cases (2) and (3). Membership in W[1] can be easily proved by a parametric reduction to the Short Turing Machine Acceptance Problem similar to the proof of Lemma 6.1.

Theorem 6.4. *Let $\mathcal{P}$ be a hereditary property on oriented graphs that either contains all trivial graphs and all acyclic tournaments or excludes a trivial graph and an acyclic tournament. Then $P(D, k, \mathcal{P})$ is fixed-parameter tractable.*

Proof. Suppose $\mathcal{P}$ excludes a trivial graph of size c_1 and an acyclic tournament of size c_2. Then $\mathcal{P}$ cannot contain any oriented graph D such that $|V(D)| \geq r(c_1, c_2)$ and is therefore finite. The problem $P(D, k, \mathcal{P})$ can then be decided in constant time.

Therefore assume that $\mathcal{P}$ contains all trivial graphs and all acyclic tournaments. If $|V(D)| \geq r(k, k)$ then, by Ramsey's theorem, D has either a trivial graph or an acyclic tournament of size k as an induced subgraph. Thus the given instance is a YES-instance. Otherwise, $|V(D)| < r(k, k)$ and we check all subsets $S \subseteq V(D)$ of size k to see whether $D[S]$ has property $\mathcal{P}$. This takes time $\binom{r(k,k)}{k} \cdot f(k)$, where $f(k)$ is the time taken to decide whether an oriented graph on k vertices has property $\mathcal{P}$. This proves that the problem $P(D, k, \mathcal{P})$ is fixed-parameter tractable. □

Theorem 6.5. *Let $\mathcal{P}$ be a hereditary property on oriented graphs that contains all trivial graphs but not all acyclic tournaments or vice versa. Then the problem $P(D, k, \mathcal{P})$ is* W[1]*-complete.*

Proof. Let $\mathcal{P}$ be a property on oriented graphs. Define a property $\mathcal{P}'$ on undirected graphs as follows: An undirected graph G satisfies $\mathcal{P}'$ if and only if the directed graph D satisfies $\mathcal{P}$, where $V(D) = V(G)$ and $A(D) = \{(u, v) \colon u < v, \{u, v\} \in E(G)\}$. Clearly G has an independent set of size k if and only if D has an independent set of size k and G has a clique of size k if and only if D has an acyclic tournament of size k. Also

1. $\mathcal{P}'$ is nontrivial and hereditary if and only if $\mathcal{P}$ is nontrivial and hereditary,
2. $\mathcal{P}'$ contains all trivial graphs but not all cliques if and only if $\mathcal{P}$ contains all trivial graphs but not all acyclic tournaments, and
3. $\mathcal{P}'$ contains all cliques but not all trivial graphs if and only if $\mathcal{P}$ contains all acyclic tournaments but not all trivial graphs.

The problem $P(G, k, \mathcal{P}')$ is W[1]-hard by Khot and Raman [86] and it is easy to see that $P(G, k, \mathcal{P}') \leq_{\text{FPT}} P(D, k, \mathcal{P})$. □

6.4 General Digraphs vs Oriented Graphs

In this section, we characterize general digraph properties for which the problem $P(D, k, \mathcal{P})$, when restricted to oriented graphs, becomes fixed-parameter tractable. In what follows, if $\mathcal{P}$ is a property on general directed

graphs then its restriction $\mathcal{P}'$ to oriented graphs is defined to be the set of all oriented graphs satisfying $\mathcal{P}$.

Corollary 6.5. *Let $\mathcal{P}$ be a nontrivial hereditary property on digraphs such that the induced subgraph problem, $P(D, k, \mathcal{P})$, is* W[1]*-complete. If $\mathcal{P}'$ is its restriction to oriented graphs then the problem $P(D, k, \mathcal{P}')$, restricted to oriented graphs, is fixed-parameter tractable if and only if either one of the following conditions is satisfied: (1) $\mathcal{P}$ contains all trivial graphs and all acyclic tournaments but not all complete symmetric digraphs, or (2) $\mathcal{P}$ contains all complete symmetric digraphs but not all trivial graphs and acyclic tournaments.*

Proof. ($\Leftarrow$) If $\mathcal{P}$ satisfies (1), then $\mathcal{P}'$ contains all trivial graphs and all acyclic tournaments. If $\mathcal{P}$ satisfies (2), then $\mathcal{P}'$ is finite. The FPT result then follows from Theorem 6.4.

($\Rightarrow$) If $\mathcal{P}$ satisfies neither (1) nor (2) of the theorem and, if the problem $P(D, k, \mathcal{P})$ is W[1]-complete, then $\mathcal{P}'$ is a nontrivial hereditary property on oriented graphs that satisfies either all trivial graphs but not all acyclic tournaments or vice versa. The hardness proof then follows from Theorem 6.5. □

Acyclic digraphs form an example of a hereditary property that contains all trivial graphs and acyclic tournaments but no complete symmetric digraphs. Consequently, the Induced Acyclic Subgraph problem is W[1]-complete on general directed graphs but in FPT on oriented graphs.

6.5 Conclusion

In this chapter we characterized hereditary properties on digraphs for which finding an induced subdigraph with k vertices in a given digraph is W[1]-complete. We first did this for general directed graphs and then for oriented graphs. We also characterized hereditary properties for which the induced subgraph problem is W[1]-complete on general directed graphs but in FPT for oriented graphs.

A related problem is the Graph Modification problem $\mathcal{P}(i, j, k)$ which asks whether a given input graph G can be 'modified' by deleting at most i vertices, j edges and adding at most k edges so that the resulting graph satisfies property $\mathcal{P}$. More formally, this problem is defined as follows: Given an undirected graph $G = (V, E)$ does there exist $V' \subseteq V$, $E' \subseteq E$ and $E'' \subseteq E^c$ (the edge set of the complement graph) with $|V'| \leq i$, $|E'| \leq j$ and $|E^c| \leq k$, such that $G - V' - E' \cup E''$ satisfies $\mathcal{P}$?

Cai [25] has shown that if a hereditary property has a finite forbidden set, the graph modification problem $\mathcal{P}(i, j, k)$ is fixed-parameter tractable with parameters i, j, k. There is no general result for the case when the forbidden

set is infinite. For instance, the Odd Cycle Transversal problem is fixed-parameter tractable [117] whereas the Wheel-Free Vertex (Edge) Deletion problem is W[2]-hard [95]. It would be interesting to obtain a dichotomy result for the infinite forbidden set case.

The graph modification problem can be framed for directed graphs as well (for directed graphs, E^c can be viewed as the set of all arcs not in input digraph D). For example, the well-known Directed Feedback Vertex Set problem can be cast as the problem $\mathcal{P}(k, 0, 0)$, where $\mathcal{P}$ is the set of all acyclic digraphs. It would be interesting to investigate the parameterized complexity of the Graph Modification problem in directed graphs.

Chapter 7

The Directed Full Degree Spanning Tree Problem

We continue our study of the parameterized complexity of NP-optimization problems in directed graphs in this chapter. Here we consider a directed analog of the Full Degree Spanning Tree problem where, given a digraph D and a non-negative integer k, the goal is to construct a spanning out-tree T of D such that at least k vertices in T have the same out-degree as in D. We show that this problem is W[1]-hard on the class of directed acyclic graphs and strongly connected digraphs. In the dual version, called Reduced Degree Spanning Tree, we parameterize from the other extreme and here the goal is to construct a spanning out-tree T such that at most k vertices in T have out-degrees that are different from that in D. We show that this problem is fixed-parameter tractable and admits a problem kernel with at most $8k$ vertices on strongly connected digraphs and $O(k^2)$ vertices on general digraphs. We also give an algorithm for this problem on general digraphs with running time $O(5.942^k \cdot n^{O(1)})$, where n is the number of vertices in the input digraph.

7.1 Problem Definition and Previous Work

The Full Degree Spanning Tree problem asks, given a connected undirected graph $G = (V, E)$ and a non-negative integer k as inputs, whether G has a spanning tree T in which at least k vertices have the same degree in T as in G. This problem was first studied by Pothof and Schut [112] in the context of water distribution networks where the goal is to determine the flow in a network by installing a small number of flow-meters. It so happens that to measure the flow in each pipe of the network, it is sufficient to find a spanning tree of the network and install flow-meters at those vertices whose degree in the spanning tree is smaller than that in the network. To find the optimal number of flow-meters (which is an expensive equipment) one needs to find a spanning tree with the largest number of vertices of full degree.

This problem has attracted a lot of attention [16, 23, 88, 72, 65]. Bhatia et al. [16] studied this problem from the point-of-view of approximation algorithms and gave a factor-$\Theta(\sqrt{n})$ algorithm for it, where n in the number of vertices in the input graph. They also showed that this problem admits no factor $O(n^{1/2-\epsilon})$ approximation algorithm unless NP = co-R. For planar graphs, a polynomial-time approximation scheme (PTAS) was presented. Independently, Broersma et al. [23] developed a PTAS for planar graphs and showed that this problem can be solved in polynomial time in special classes of graphs such as bounded treewidth graphs and co-comparability graphs. Guo et al. [72] studied the parameterized complexity of this problem and showed it to be W[1]-hard. Gaspers et al. [65] give an $O(1.9465^n \cdot n^{O(1)})$ algorithm for the optimization version of this problem.

One can parameterize the d-FDST problem from the "other end" and ask whether a graph G has spanning tree T in which at most k vertices have degrees different from that in G. This problem has been studied under the name Vertex Feedback Edge Set and is defined as follows. Given a connected undirected graph $G = (V, E)$ and a nonnegative integer k, find an edge subset E' incident on at most k vertices such that $G[E \setminus E']$ is acyclic. Note that if there exists such an edge set E', then there exists $E'' \subseteq E'$ such that $G[E \setminus E'']$ is a spanning tree. Khuller et al. [88] show that this problem is MAX SNP-hard and describe a $(2+\epsilon)$-approximation algorithm for it for any fixed $\epsilon > 0$. Guo et al. [72] show that this problem is fixed-parameter tractable by demonstrating a problem kernel with at most $4k$ vertices.

Here we consider a natural generalization of these problems to directed graphs. An *oriented tree* is a tree in the undirected sense each of whose edges has been assigned a direction. We say that a subdigraph T of a directed graph $D = (V, A)$ is an *out-tree* if it is an oriented tree with exactly one vertex s of in-degree zero (called the *root*). An out-tree that contains all vertices of D is an *out-branching* of D. Given a digraph $D = (V, A)$ and an out-tree T of D, we say that a vertex $v \in V$ is of *full degree* if its out-degree in T is the same as that in D; otherwise, v is said to be of *reduced degree*. We define the Directed Full Degree Spanning Tree (d-FDST) problem as follows.

Input: Given a directed graph $D = (V, A)$ and an integer k.

Parameter: The integer k.

Question: Does there exist an out-branching of D in which at least k vertices are of full degree?

We call the dual of this problem the Directed Reduced Degree Spanning Tree (d-RDST) problem.

Input: Given a directed graph $D = (V, A)$ and an integer k.

Parameter: The integer k.

Question: Does there exist an out-branching of D in which at most k vertices are of reduced degree?

We show that d-FDST is W[1]-hard, by a reduction from the Independent Set problem, for two important digraph classes: directed acyclic graphs (DAGs) and strongly connected digraphs. We show that d-RDST is fixed-parameter tractable (FPT) by exhibiting a problem kernel with at most $O(k^2)$ vertices. For strongly connected digraphs, d-RDST admits a kernel with at most $8k$ vertices. We also develop an algorithm for d-RDST with running time $O(5.942^k \cdot n^{O(1)})$.

Related Results. The Full Degree Spanning Tree problem is one of the many variants of the generic Constrained Spanning Tree problem, where one is required to find a spanning tree of a given (di)graph subject to certain constraints. This class of problems has been studied intensely [3, 37, 41, 52, 58, 73, 113].

In [52], the authors consider the problem Max Leaf Spanning Tree where one is required to find a spanning tree of an undirected graph with the maximum number of leaves. When parameterized by the solution size, this problem admits a kernel with $3.75k$ vertices. In the directed variant of this problem, one has to decide whether an input digraph D has an out-branching with at least k leaves. This problem admits a kernel with $O(k^3)$ vertices, provided the root of the out-branching is *given as part of the input* [58], and has an algorithm with run-time $O(3.72^k \cdot n^{O(1)})$ [41]. Another such problem is Max Internal Spanning Tree, where the objective is to find a spanning tree (or an out-branching, in case of digraphs) with at least k internal vertices. For undirected graphs, a $3k$-vertex kernel and an algorithm with running time $O(8^k \cdot n^{O(1)})$ is known for this problem [61]. For directed graphs, an $O(k^2)$-vertex kernel due to [73] and an algorithm with running time $O(40^k \cdot n^{O(1)})$ due to [37] is known.

Organization of this Chapter. In Section 7.2 we define the relevant notions related to digraphs. In Section 7.3 we show that d-FDST is W[1]-hard on two classes of digraphs: directed acyclic graphs and strongly connected digraphs. In Section 7.4 we show that the d-RDST problem is in FPT by demonstrating a kernel with at most $O(k^2)$ vertices. We first demonstrate a kernel with $8k$ vertices for strongly connected digraphs and use the ideas therein to develop the $O(k^2)$ kernel for general digraphs. In Section 7.5 we develop an algorithm for the d-RDST problem with running time $O(5.942^k \cdot n^{O(1)})$. Finally in Section 7.6, we end with some concluding remarks and open questions.

7.2 Digraphs: Basic Terminology

The notation and terminology that we follow are from [12]. Given a digraph D we let $V(D)$ and $A(D)$ denote the vertex set and arc set, respectively, of D. If $u, v \in V(D)$, we say that u is an *in-neighbour* (*out-neighbour*) of v if $(u, v) \in A(D)$ ($(v, u) \in A(D)$). The in-degree $d^-(u)$ (out-degree $d^+(u)$) of u is the number of in-neighbours (out-neighbours) of u. Given a subset $V' \subseteq V(D)$, we let $D[V']$ denote the digraph induced on V'. The *underlying undirected graph* $U(D)$ is the undirected graph obtained from D by disregarding the orientation of arcs and deleting an edge for each pair of parallel edges in the resulting graph. The *connectivity components* of D are the subdigraphs induced by the vertices of components of $U(D)$.

A digraph is *oriented* if every pair of vertices has at most one arc between them. A (v_1, v_s)-*walk* in $D = (V, A)$ is a sequence $v_1, \ldots, v_s$ of vertices such that $(v_i, v_{i+1}) \in A$ for all $1 \leq i \leq s-1$. A *dicycle* is a walk $v_1, v_2, \ldots, v_s$ such that $s \geq 3$, the vertices $v_1, \ldots, v_{s-1}$ are distinct and $v_1 = v_s$. A digraph with no dicycles is called a *directed acyclic graph (DAG)*. A digraph D is *strongly connected* if for every pair of distinct vertices $u, v \in V(D)$, there exists a (u, v)-walk and a (v, u)-walk. A *strong component* of a digraph is a maximal induced subdigraph that is strongly connected. The *strong component digraph* $SC(D)$ is the directed acyclic graph obtained by contracting each strong component to a single vertex and deleting any parallel arcs obtained in this process. A strong component S of a digraph D is a *source strong component* if no vertex in S has an in-neighbour in $V(D) \setminus V(S)$. The following is a necessary and sufficient condition for a digraph to have an out-branching.

Proposition 7.1 ([12])**.** *A digraph D has an out-branching if and only if D has a unique source strong component.*

One can obtain the strongly connected components of a digraph D in time $O(n + m)$ [38], where $n = |V(D)|$ and $m = |A(D)|$. Each strong component can be stored as an n-bit vector where the ith bit is a one if and only if vertex i is in the strong component. It is now easy to see that one can verify in $(n + m)$ time whether there exists a unique source strong component.

7.3 The d-FDST Problem

We now show that d-FDST is W[1]-hard on two important digraph classes: DAGs and strongly connected digraphs. This is a modification of the reduction presented in [88] (Lemma 3.2).

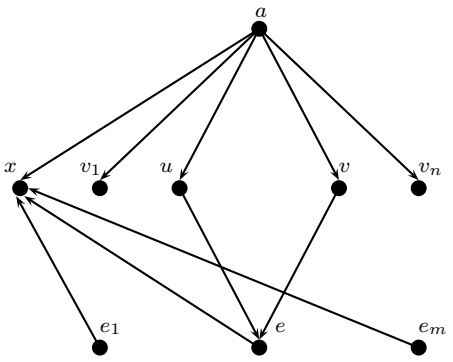


Figure 7.1: The digraph D.

Theorem 7.1. *The d-FDST problem parameterized by the solution size is* W[1]*-hard on directed acyclic graphs (DAGs) and strongly connected digraphs. Also the d-RDST problem is* NP*-hard on strongly connected digraphs.*

Proof. We show that k-Independent Set, which is known to be W[1]-complete [47], fixed-parameter reduces to the d-FDST problem. Let (G, k) be an instance of the k-Independent Set problem where we assume G to be a *connected* undirected graph on n vertices and m edges. Construct a directed graph D as follows. The vertex set $V(D)$ consists of $n + m + 2$ vertices: $v_1, \ldots, v_n, e_1, \ldots, e_m, a, x$, where the vertices v_i, for $1 \leq i \leq n$, and e_j, for $1 \leq j \leq m$, "correspond", respectively, to the vertices and edges of G and a, x are two special vertices. The digraph D can be viewed as a three-layer graph. Layer one consists of vertex a. Layer two consists of the vertices $x, v_1, \ldots, v_n$ and vertex a has an out-arc to each vertex in layer two. Layer three consists of the vertices $e_1, \ldots, e_m$ and each e_j, for $1 \leq j \leq m$, has an out-arc to vertex x. If $e = \{u, v\} \in E(G)$ then the vertices u and v in layer two have an out-arc each to vertex e in layer three. This completes the description of D. It is easy to verify that D is a DAG.

Observe that in any out-branching T of D every vertex (except the root) has exactly one in-neighbor and that:

1. Vertices $a, e_1, \ldots, e_m$ are the only vertices of D of in-degree zero and therefore the root of T must be one of these. Moreover, if a preserves its out-degree in T then vertices $e_1, \ldots, e_m$ must be of reduced degree.

2. At most one vertex from among $e_1, \ldots, e_m$ can preserve its out-degree in T because each of them has an out-arc to x.

3. Vertex x preserves its out-degree in T because x is of out-degree zero.

We claim that G has an independent set of size k if and only if the digraph D has an out-branching with $k+2$ vertices of full degree. Suppose that G has a k-independent set on the vertices $v_{i_1}, \ldots, v_{i_k}$. Consider the subdigraph T' induced by the vertices $a, v_{i_1}, \ldots, v_{i_k}$ and their out-neighbors. It is easy to verify that T' is actually an out-tree rooted at a in which vertices $a, x, v_{i_1}, \ldots, v_{i_k}$ have full degree. For each edge $e \in E(G)$ that is not incident to any vertex in the k-independent set, arbitrarily choose one of its endpoints, say v, and add the arc (v, e) to the out-tree T'. This converts the out-tree T' into an out-branching T with at least $k+2$ vertices of full degree. Conversely suppose that D admits an out-branching T in which at least $k+2$ vertices preserve their out-degree. We consider two cases.

Case 1. Vertex a preserves its out-degree. Then no vertex from layer three preserves its out-degree, as each of these vertices has an out-arc to x. Since x is the other vertex of full degree, it must be that k vertices from among $v_1, \ldots, v_n$ preserve their out-degree. No pair from among these k vertices form an edge in G, for otherwise, they would have an out-arc to the same vertex e in layer three in T and this would contradict the assumption that T is an out-branching. Hence these k vertices must be independent in G.

Case 2. Vertex a does not preserve its out-degree. By Observation 2, at most one vertex from layer three can preserve its out-degree and, by Observation 3, x preserves its out-degree in every out-branching. Hence at least k vertices from among the $v_1, \ldots, v_n$ preserve their out-degree. These vertices from a k-independent set in G. This shows that d-FDST is W[1]-hard on DAGs.

By modifying the above reduction from k-Independent Set, we show that d-FDST is W[1]-hard on the class of strongly connected digraphs (and hence that the d-RDST problem is NP-hard on this class of digraphs). Given an instance (G, k) of k-Independent Set, it is no loss of generality to assume that G is a *connected non-bipartite* graph. Construct the digraph D as above with just one modification: add the arc (x, a). It is easy to verify that the resulting digraph is strongly connected. Then G has an independent set of size k if and only if D admits an out-branching with $k+2$ vertices of full degree. Suppose G has a k-independent set on the vertex set $\{v_{i_1}, \ldots, v_{i_k}\}$. Since G is non-bipartite, there exists an edge e_j both of whose endpoints are in $V(G) \setminus \{v_{i_1}, \ldots, v_{i_k}\}$. It is easy to see that there is an out-branching with e_j as root in which the vertices $e_j, x, v_{i_1}, \ldots, v_{i_k}$ are of full degree. Conversely suppose that D has an out-branching with $k+2$ vertices of full degree. Between a and x, at most one can preserve its out-degree and among $a, e_1, \ldots, e_m$ at most one can preserve its out-degree. Therefore at least k vertices from among $v_1, \ldots, v_n$ preserve their out-degree. These vertices form an independent set in G. This shows that d-RDST is NP-hard on strongly connected digraphs and completes the proof of the theorem. □

7.4 d-RDST: A Problem Kernel

In this section we show that d-RDST admits a problem-kernel with $O(k^2)$ vertices and is therefore fixed-parameter tractable. We first consider the special case when the input digraph is strongly connected and establish a kernel with $8k$ vertices for this case. This will give some insight as to how to tackle the general case.

Observe that if (D, k) is a YES-instance of the d-RDST problem and T is a solution out-branching (one in which at most k vertices are of reduced out-degree), then the subdigraph of D induced by the vertices of full degree is a forest in the undirected sense and hence has treewidth one (for more on treewidth, see [20, 91]). Therefore the underlying undirected graph $U(D)$ has treewidth at most $k + 1$. Moreover one can show that the property of having an out-branching with at most k vertices of reduced out-degree is expressible in monadic second-order logic [39]. One can now use the results of Arnborg et al. [7] to conclude that for every fixed k the d-RDST problem can be decided in linear time. This shows that the d-RDST problem is fixed-parameter tractable. However the running time dependence of this algorithm on k is huge making it impractical. In what follows, we give an alternative algorithm with a more well-behaved dependence on the parameter k.

7.4.1 A Linear Kernel for Strongly Connected Digraphs

We actually establish the $8k$-vertex kernel for a more general class of digraphs, those in which every vertex has out-degree at least one. Call this class of digraphs *out-degree at least one digraphs* and denote it by $\mathscr{D}_1^+$. It is easy to see that strongly connected digraphs (SCDs) is a subclass of $\mathscr{D}_1^+$. Since a digraph in $\mathscr{D}_1^+$ can have vertices of in-degree zero, it follows that SCDs form a proper subclass of $\mathscr{D}_1^+$.

There are three simple reduction rules for the case where the input is an out-degree at least one digraph. We assume that the input is (D, k).

Rule 1. If there exists $u \in V(D)$ such that $d^-(u) \geq k + 2$ then return NO; else return (D, k).

Rule 2. If there are $k + 1$ vertices of out-degree at least $k + 1$ then return NO; else return (D, k).

Rule 3 (The Path Rule). Let $x_0, x_1 \ldots, x_{p-1}, x_p$ be a sequence of vertices in D such that for $0 \leq i \leq p - 1$ we have $d^+(x_i) = 1$ and $(x_i, x_{i+1}) \in A(D)$. Let Y_0 be the set of in-neighbours of $x_1, \ldots, x_{p-1}$ and let $Y := Y_0 \setminus \{x_0, x_1, \ldots, x_{p-2}\}$. Delete the vertices $x_1, \ldots, x_{p-1}$ and add two new vertices z_1, z_2 and the arcs $(x_0, z_1), (z_1, z_2), (z_2, x_p)$. For $y \in Y$, if y has at least two out-neighbors in $\{x_1, \ldots, x_{p-1}\}$,

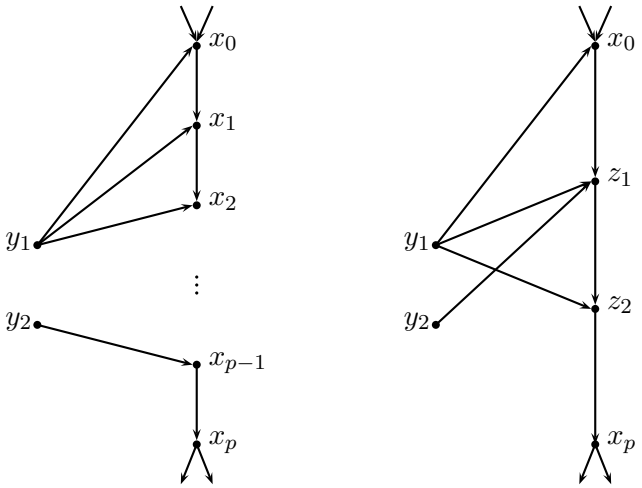


Figure 7.2: Illustrating the *Path Rule*: the left and right-hand sides show, respectively, the situation before and after the transformation. Vertex y_1 has two neighbors and vertex y_2 just one neighbor in the set $\{x_1, \ldots, x_{p-1}\}$.

then add arcs $(y, z_1), (y, z_2)$; if y has exactly one out-neighbor in $\{x_1, \ldots, x_{p-1}\}$, then add the arc (y, z_1). Return (D, k). See Figure 7.2.

It is easy to see that Rules 1 and 2 are indeed reduction rules for the d-RDST problem on out-degree at least one digraphs. If a vertex v has in-degree at least $k + 2$ then at least $k + 1$ in-neighbors of u must be of reduced degree in any out-branching. This shows that Rule 1 is a reduction rule. If a vertex u has out-degree $k + 1$ and is of full degree in some out-branching T then T has at least $k+1$ leaves. Since the input digraph is such that every vertex has out-degree at least one, this means that in T there are at least $k + 1$ vertices of reduced degree. This shows that any vertex of out-degree $k + 1$ must necessarily be of reduced degree in any solution out-branching. Therefore if there are $k + 1$ such vertices, the given instance is a NO-instance. This proves that Rule 2 is a reduction rule.

Lemma 7.1. *Rule 3 is a reduction rule for the d-RDST problem.*

Proof. It is sufficient to show that if (D', k) is the instance obtained by one application of Rule 3 to an instance (D, k), then D has an out-branching with at most k vertices of reduced out-degree if and only if D' has an out-branching with at most k vertices of reduced degree.

Suppose D' has an out-branching T' with at most k vertices of reduced degree. There are two cases to consider. In the first case, there are no arcs from Y to z_1 or z_2 in T'. In this case we may assume without loss of generality that the path $x_0 \to z_1 \to z_2$ occurs as a sub-path of T'. For if $x_0 \to z_1 \to z_2$ is not a subpath of T', then one of z_1 or z_2 has in-degree

zero in T'. Hence it must be that either z_1 or z_2 is the root of T'. If z_1 is the root of T' then x_0 is a leaf in T'; if z_2 is the root then z_1 is a leaf. In either case, we can make x_0 the root and maintain the path $x_0 \to z_1 \to z_2$ without increasing the number of vertices of reduced degree. In order to construct an out-branching T for D, replace $x_0 \to z_1 \to z_2$ by the path $x_0 \to x_1 \to \cdots \to x_{p-1}$. Moreover if T' contains the arc (z_2, x_p) then, in constructing T, add the arc (x_{p-1}, x_p). Note that T has at most k vertices of reduced degree.

In the second case, there exists at least one vertex $y \in Y$ with arcs to $\{z_1, z_2\}$. Suppose that T' contains the arcs $(y_1, z_1), (y_2, z_2)$, where y_1 and y_2 are (not necessarily distinct) vertices in Y . Note that both x_0 and z_1 are of out-degree zero in T' and hence of reduced degree. Observe that $T' \setminus \{z_1\}$ is an out-branching for $D' \setminus \{z_1\}$ as z_1 is a leaf in T'. We transform T' into another out-branching for D' by deleting the arc (y_1, z_1) and adding the arc (x_0, z_1). In this new out-branching, x_0 is of full degree and y_1 is of reduced degree but the number of vertices of reduced degree does not increase.

We can therefore assume without loss of generality that in T' there is exactly one vertex $y \in Y$ with an out-arc to $\{z_1, z_2\}$. Suppose $(y, z_2) \in A(T')$. Then y must be of reduced degree as whenever we have an arc (y, z_2), we also have an arc (y, z_1). In this case we transform T' by deleting the arcs $(y, z_2), (x_0, z_1)$ and introducing the arcs $(y, z_1), (z_1, z_2)$. The resulting digraph is an out-branching with at most k vertices of reduced degree as x_0 now is of reduced degree but z_1 is of full degree. Therefore we are left to consider the case when y has an arc to z_1 only. Let x_s be the first out-neighbor of y in $\{x_1, \ldots, x_{p-1}\}$. Delete z_1, z_2 and connect x_0 to the dipath $x_1 \to \cdots \to x_{s-1}$ and y to the dipath $x_s \to \cdots \to x_{p-1}$. Add the arc (x_{p-1}, x_p) if $(z_2, x_p) \in A(T')$. The resulting digraph is an out-branching for D with at most k vertices of reduced degree.

To prove the converse, suppose that D has an out-branching T with at most k vertices of reduced degree. Again there are two cases to consider.

Case 1. There are no arcs from Y to any x_i, for $1 \leq i \leq p-1$, in T. There are two sub-cases here. Either T contains the dipath $x_0 \to x_1 \to \cdots x_{p-1} \to x_p$, in which case we can compress it to the path (x_0, z_1, z_2, x_p) to obtain an out-branching T' for D' with at most k vertices of reduced degree. Otherwise one of the vertices $x_1, \ldots, x_p$ must be the root of T. If x_p is the root, then T contains the dipath $x_0 \to x_1 \to \cdots x_{p-1}$ and we replace it by (x_0, z_1, z_2) to obtain an out-branching T' of D'. If one of $x_1, \ldots, x_{p-1}$ is the root, then delete $x_1, \ldots, x_{p-1}$, make z_1 the root and add the arcs $(z_1, z_2), (z_2, x_p)$. This transforms T into an out-branching of D' with at most k vertices of reduced degree.

Case 2. Now suppose that in T the vertices $y_{i_1}, \ldots, y_{i_s} \in Y$ have out-neighbors in $x_1, \ldots, x_{p-1}$. Since T is an out-branching, the set of out-neighbors of y_{i_j} and y_{i_l} are disjoint for all $j \neq l$. In T, the out-neighbors

of y_{i_j} in $\{x_1, \ldots, x_{p-1}\}$ can be ordered in the natural way according to their position in the path $x_1 \to \cdots \to x_{p-1}$. Let x_{q_j} be the first out-neighbor of y_{i_j} among $\{x_1, \ldots, x_{p-1}\}$ in T. Transform T into a digraph T_1 by deleting out-arcs such that for $1 \leq j \leq s$, the only out-neighbor of y_{i_j} among $\{x_1, \ldots, x_{p-1}\}$ is x_{q_j}. Sort the vertices $y_{i_1}, \ldots, y_{i_s}$ in increasing order based on the order of the vertices x_{q_j} in the path $x_1 \to \cdots \to x_{p-1}$. Without loss of generality we assume that the sorted order is also $y_{i_1}, \ldots, y_{i_s}$.

Transform T_1 yet again by connecting the vertices x_i so that the resulting digraph (which we continue to call T_1) contains the paths:

- $x_0 \to x_1 \to \cdots \to x_{q_1-1}$;
- $y_{i_j} \to x_{q_j} \to \cdots \to x_{q_{j+1}-1}$, for $1 \leq j \leq s-1$;
- $y_{i_s} \to x_{q_s} \to \cdots \to x_{p-1} \cdots$.

Note that in T, the vertices x_{q_j-1}, for $1 \leq j \leq s$, are of out-degree zero. Moreover the last path $y_{i_s} \to x_{q_s} \to \cdots$ in the sequence contains all vertices $x_{q_s}, \ldots, x_{p-1}$ but may or may not contain the vertex x_p. Observe that T_1 is an out-tree and that the only vertices $y \in \{y_{i_1}, \ldots, y_{i_s}\}$ whose out-degree is reduced in this transformation had at least two out-neighbors among the vertices $\{x_1, \ldots, x_{p-1}\}$ in T. Hence for every y_{i_j} whose out-degree is reduced in transforming T to T_1, there exists a distinct vertex in $x_1, \ldots, x_{p-1}$ of out-degree zero in T which is of full degree in T_1. Thus the number of vertices of reduced degree does not change in this transformation.

Now delete the arcs (y_{i_j}, x_{q_j}), where $1 \leq j \leq s-1$, and add (x_{q_j-1}, x_{q_j}), where $1 \leq j \leq s-1$, so that the resulting digraph T_2 contains the path $x_0 \to \cdots \to x_{q_{s-1}}$. In this transformation, the vertices which possibly have their out-degree reduced are y_{i_j}, for $1 \leq j \leq s-1$, but an equal number of vertices x_{q_j-1}, for $1 \leq j \leq s-1$, attain full degree. Therefore the number of vertices of reduced out-degree in T_2 is at most that in T_1. To obtain an out-branching of D' from T_2, delete $x_1, \ldots, x_{q_{s-1}}$, add the arcs (y_{i_s}, z_1), (z_1, z_2) and connect z_2 to the out-neighbor of x_{p-1}, if any. Note that this transforms T_2 into an out-branching of D' with at most k vertices of reduced degree.

This completes the proof of the lemma. □

It is easy to see that Rules 1 and 2 can be applied in $O(n)$ time and that Rule 3 can be applied in $O(n+m)$ time. Note that Rule 3 is *parameter independent*, that is, an application of the rule does not affect the parameter. Consequently, it makes sense to talk about a digraph being reduced with respect to Rule 3 as distinct from an instance of d-RDST being reduced with respect to Rule 3. Our kernelization algorithm consists in applying Rules 1 to 3 repeatedly until the given instance is reduced.

We next describe a lemma that we repeatedly make use of in the sequel. Given a directed graph D, we let $V_i(D) \subseteq V(D)$ denote the set of vertices of out-degree i; $V_{\geq i}(D) \subseteq V(D)$ denotes the set of vertices of out-degree at least i.

Lemma 7.2. *Let D be a directed graph reduced with respect to the Path Rule (Rule 3) and let T be an out-branching of D with root r such that X is the set of vertices of reduced out-degree. Then*

$$|V(T)| \leq 4|V_0(T) \cup V_{\geq 2}(T) \cup X| \leq 4(|V_0(T)| + |X \cup V_0(T)|).$$

Proof. If we view the out-branching T as an undirected graph, $V_0(T)$ is the set of leaves and $V_{\geq 2}(T)$ is the set of vertices of degree at least three along with the root r, if $d^+_T(r) \geq 2$. Thus $V_{\geq 2}(T)$ has at most one vertex of total degree two and all other vertices are of total degree at least three. It is a well-known fact that a tree with l leaves has at most $l-1$ internal vertices of degree at least three. Since $V_{\geq 2}(T)$ has at most one vertex of total degree two, we have $|V_{\geq 2}(T)| \leq |V_0(T)|$.

Now consider the vertices of the out-branching T which have out-degree exactly one. Define $W := X \cup V_0(T) \cup V_{\geq 2}(T)$ and let $\mathcal{P}$ be the set of maximal dipaths in T such that for any dipath $P = x_0 \to x_1 \to \cdots \to x_p$ in $\mathcal{P}$ we have that (1) $d^+_D(x_i) = 1$ for $0 \leq i \leq p-1$, and (2) $x_p \in W$. Observe that every vertex with out-degree exactly one in T is contained in exactly one path in $\mathcal{P}$. Also observe that the set of vertices of out-degree exactly one in T *not* contained in W is precisely the set $V_1(T) \setminus X$. Therefore $|V_1(T) \setminus X| \leq \sum_{P \in \mathcal{P}} (|P| - 1)$, where $|P|$ denotes the number of *vertices* in the path P. By Rule 3, any dipath $P \in \mathcal{P}$ has at most four vertices and since the number of dipaths in $\mathcal{P}$ is at most $|W|$, we have

$$|V_1(T) \setminus X| \leq 3 \cdot |\mathcal{P}| \leq 3 \cdot |W| \leq 3|X \cup V_0(T) \cup V_{\geq 2}(T)|.$$

Since $|V(T)| \leq |V_1(T) \setminus X| + |X \cup V_0(T) \cup V_{\geq 2}(T)|$, we have

$$|V(T)| \leq 4|V_0(T) \cup V_{\geq 2}(T) \cup X| \leq 4(|V_0(T)| + |V_0(T) \cup X|).$$

This completes the proof of the lemma. □

We can now bound the size of a yes-instance of the d-RDST problem on out-degree at least one digraphs that have been reduced with respect to Rules 1 to 3.

Theorem 7.2. *Let (D, k) be a yes-instance of the d-RDST problem on out-degree at least one digraphs that is reduced with respect to Rules 1 to 3. Then $|V(D)| \leq 8k$.*

Proof. Since (D, k) is a YES-instance of the problem, let T be an out-branching of D and let X be the set of vertices of reduced degree in T, where $|X| \leq k$. Every vertex of D is of out-degree at least one and hence $V_0 \subseteq X$, where V_0 is the set of leaves in T. Consequently $|X \cup V_0| \leq k$ and $|V_0| \leq k$ and by Lemma 7.2, we have $|V(T)| \leq 8k$, as claimed. □

Observe that the crucial step in the proof above was to bound the number of leaves in the solution out-branching. For out-degree at least one digraphs this is easy since every leaf is a vertex of reduced degree. This is not the case with general digraphs which may have an arbitrary number of vertices of out-degree zero, all of which are of full degree in any out-branching. In the next subsection we present a set of reduction rules for the d-RDST problem in general digraphs which help us bound the number of vertices of out-degree zero in terms of the parameter k.

7.4.2 An $O(k^2)$-Vertex Kernel in General Digraphs

For general digraphs, we first consider an *annotated version* of the problem as this seems to help in developing reduction rules. Eventually we will revert to the original unannotated version. An instance of the annotated version consists of a triplet (D, X, k) where D and k are, respectively, the input digraph and the parameter, and X is a subset of $V(D)$ such that in *any* out-branching with at most $|X| + k$ vertices of reduced degree, the vertices of X must be of reduced degree. The question in this case is to decide whether D admits an out-branching where the set of vertices of reduced degree is $X \cup S$, where $S \subseteq V(D) \setminus X$ and $|S| \leq k$. Call such an out-branching a *solution out-branching*. To obtain a kernel for d-RDST, we apply the reduction rules to an instance (D, k) after setting $X = \emptyset$.

Given an instance (D, X, k), we define the *conflict set* of a vertex $u \in V(D) \setminus X$ as

$$C(u) := \{v \in V(D) \setminus X : N^+(u) \cap N^+(v) \neq \emptyset\}.$$

Clearly vertices of out-degree zero have an empty conflict set. If a vertex v has a non-empty conflict set then in any out-branching either v loses its degree or *every* vertex in $C(v)$ loses its degree. Moreover if $u \in C(v)$ then $v \in C(u)$ and in this case we say that u and v are in conflict. The *conflict number* of D is defined as $c(D) := \sum_{v \in V(D) \setminus X} |C(v)|$.

We assume that the input instance is (D, X, k) and the kernelization algorithm consists in applying each reduction rule repeatedly, in the order given below, until no longer possible. Therefore when we say that Rule i is indeed a reduction rule we assume that the input instance is reduced with respect to the rules preceding it.

Rule 0. If $u \in X$ and $d^+(u) = 1$, delete the out-arc from u and return (D, X, k).

The vertices in X are of reduced degree in any solution out-branching. Thus if a vertex in X has out-degree exactly one, this out-arc will never be part of a solution out-branching, and deleting it will not change the solution structure.

Rule 1. If there exists $u \in V(D)$ such that the number of in-neighbors of u in $V(D) \setminus X$ is at least $k+2$ then return NO; else return (D, X, k).

In the last subsection, we already showed that this rule is indeed a reduction rule.

Rule 2. If $u \in V(D) \setminus X$ and $|C(u)| > k$, set $X \leftarrow X \cup \{u\}$ and $k \leftarrow k-1$. Furthermore if $d^+(u) = 1$ then delete the out-arc from u and return (D, X, k).

If the conflict set $C(u)$ of $u \in V(D) \setminus X$ is of size at least $k+1$ and if u is of full degree in some out-branching T, then every vertex in $C(u)$ must be of reduced degree in T. Therefore if (D, X, k) is a YES-instance then u must have its degree reduced in *every* solution out-branching. In addition, if u has out-degree exactly one, then this out-arc cannot be part of any solution out-branching and can be deleted. This shows that Rule 2 is a reduction rule.

Rule 3. If $c(D) > 2k^2$ then return NO, else return (D, X, k).

Lemma 7.3. *Rule 3 is a reduction rule for the d-RDST problem.*

Proof. To see why Rule 3 qualifies to be a reduction rule, construct the *conflict graph* $\mathcal{C}_{D,X}$ of the instance (D, X, k) which is defined as follows. The vertex set $V(\mathcal{C}_{D,X}) := V(D) \setminus X$ and two vertices in $V(\mathcal{C}_{D,X})$ have an edge between them if and only if they are in conflict. Since the size of the conflict set of any vertex is at most k (as D is reduced with respect to Rule 2), the degree of any vertex in $\mathcal{C}_{D,X}$ is at most k. The key observation is that if T is any solution out-branching of (D, X, k) in which the set of vertices of reduced degree is $X \cup S$ with $S \subseteq V(D) \setminus X$, then S forms a vertex cover of $\mathcal{C}_{D,X}$. Since we require that $|S| \leq k$, the number of edges in $\mathcal{C}_{D,X}$ is at most k^2. For a vertex $v \in V(D) \setminus X$, let $d'(v)$ be the number of neighbors of vertex v in the conflict graph $\mathcal{C}_{D,X}$. Observe that $c(D) := \sum_{v \in V(D) \setminus X} |C(v)| = \sum_{v \in V(D) \setminus X} d'(v) \leq 2k^2$. The last inequality follows from the fact that sum of degrees of vertices in a graph is equal to twice the number of edges. □

Rule 4. If $u \in V(D)$ such that $d^+(u) = 0$ and $d^-(u) = 1$ then delete u from D and return (D, X, k).

It is easy to see that Rule 4 is a reduction rule: vertex u is of full degree in any solution and it does not determine whether its parent is of full or reduced degree in a solution out-branching and therefore can be deleted. To obtain a solution out-branching for D from a solution T' for $D \setminus v$, simply add the arc between u and its parent in T'.

Rule 5. Let $u \in V(D)$ be of out-degree zero and let $v_1, \ldots, v_r$ be its in-neighbors, where $r > 2$. Delete u and add $\binom{r}{2}$ new vertices u_{ij}, where $1 \leq i < j \leq r$; for a newly added vertex u_{ij} add the arcs (v_i, u_{ij}) and (v_j, u_{ij}). Return (D, X, k).

Note that vertex u forces at least $r-1$ vertices from $\{v_1, \ldots, v_r\}$ to be of reduced degree in any out-branching of D. This situation is captured by deleting u and introducing $\binom{r}{2}$ vertices as described in the rule. These $\binom{r}{2}$ vertices then force at least $r-1$ vertices from $\{v_1, \ldots, v_r\}$ to be of reduced degree in any out-branching of the transformed graph. The upshot is that each vertex of out-degree zero has in-degree exactly two.

Lemma 7.4. *Rule 5 is a reduction rule for the d-RDST problem.*

Proof. Let (D, X, k) and (D', X, k) be the instances of d-RDST before and after one application of Rule 5, respectively. We claim that (D, X, k) is a YES-instance if and only if (D', X, k) is a YES-instance.

Let T be an out-branching of D that certifies that (D, X, k) is a YES-instance. Then at least $r-1$ vertices from $\{v_1, \ldots, v_r\}$ are of reduced degree in T. Transform T into an out-branching T' for (D', X, k) as follows. Delete u from T and introduce the vertices u_{ij} for $1 \leq i < j \leq r$. If $v_i \in \{v_1, \ldots, v_r\}$ was of full degree in T then in T' add the arcs (v_i, u_{pq}) for all $1 \leq p < q \leq r$; otherwise add the arcs (v_1, u_{pq}) for all $1 \leq p < q \leq r$. The out-branching T' certifies that (D', X, k) is a YES-instance.

Conversely suppose that the out-branching T' certifies that (D', X, k) is a YES-instance. Again at least $r-1$ vertices from $\{v_1, \ldots, v_r\}$ are of reduced degree in T'. Transform T' into an out-branching T for (D, X, k) as follows. Delete the vertices u_{ij} for $1 \leq i < j \leq r$ and introduce vertex u. If $v_i \in \{v_1, \ldots, v_r\}$ was of full degree in T', add the arc (v_i, u) in T; otherwise add the arc (v_1, u). Clearly T certifies that (D, X, k) is a YES-instance. □

Rule 6. If $u, v \in V(D) \setminus X$ have $p > 1$ common out-neighbors of out-degree zero, delete all but one of them. Return (D, X, k).

Rule 7. If $u \in V(D)$ is of out-degree zero such that at least one in-neighbor of u is in X, delete u. Return (D, X, k).

By Rule 5, it is clear that if $u, v \in V(D) \setminus X$ have at least two common out-neighbors of out-degree zero then these out-neighbors have in-degree exactly two. It is intuitively clear that these out-neighbors are equivalent in

some sense and it suffices to preserve just one of them. It is easy to show that the original instance has a solution out-branching if and only if the instance obtained by one application of Rule 6 has a solution out-branching. As for Rule 7, if u has two in-neighbors v and w and if $v \in X$, we can delete the arc (v, u) without altering the solution structure. But then v is a private neighbor of w of out-degree zero and hence can be deleted by Rule 4.

Rule 8 (The Path Rule). Let $x_0, x_1 \ldots, x_{p-1}, x_p$ be a sequence of vertices in D such that for $0 \leq i \leq p-1$ we have $d^+(x_i) = 1$ and $(x_i, x_{i+1}) \in A(D)$. Let Y_0 be the set of in-neighbours of $x_1, \ldots, x_{p-1}$ and let $Y := Y_0 \setminus \{x_0, x_1, \ldots, x_{p-2}\}$. Delete the vertices $x_1, \ldots, x_{p-1}$ and add two new vertices z_1, z_2 and the arcs $(x_0, z_1), (z_1, z_2), (z_2, x_p)$. For $y \in Y$, if y has at least two out-neighbors in $\{x_1, \ldots, x_{p-1}\}$, then add arcs $(y, z_1), (y, z_2)$; if y has exactly one out-neighbor in $\{x_1, \ldots, x_{p-1}\}$, then add the arc (y, z_1). Return (D, k).

This is Rule 3 from the previous subsection where it was shown to be a reduction rule for the d-RDST problem (note that the proof of Lemma 7.1 did not use the fact that the input was an out-degree at least one digraph). By Rule 0, no vertex on the path $x_0, x_1, \ldots, x_{p-1}$ is in X and therefore the proof of Lemma 7.1 continues to hold for the annotated case as well.

It is easy to see that a single application of Rules 5 or 6 takes time $O(n^2)$; all other rules take time $O(n+m)$. We are now ready to bound the number of vertices of out-degree zero in a reduced instance of the annotated problem.

Lemma 7.5. *Let (D, X, k) be a yes-instance of the annotated d-RDST problem that is reduced with respect to Rules 0 through 8 mentioned above. Then the number of vertices of out-degree zero in D is at most k^2.*

Proof. Let u be a vertex of out-degree zero. By Rules 4 and 5, it must have exactly two in-neighbors, say, x and y. By Rule 7, neither x nor y is in X and are therefore still in conflict in the reduced graph. Hence, either x or y must be of reduced degree in any solution out-branching. Furthermore any vertex not in X can have at most k out-neighbors of out-degree zero since, by Rule 2, any vertex not in X is in conflict with at most k other vertices and, by Rule 6, two vertices in conflict can have at most one common out-neighbor of out-degree zero. Since (D, X, k) is assumed to be a YES-instance, at most k vertices can lose their out-degree in any solution out-branching. Moreover, by Rule 4, any vertex of out-degree zero is an out-neighbor of at least one vertex of reduced degree. Therefore the total number of vertices of out-degree zero is at most k^2. □

Lemma 7.6. *Let (D, k) be a* YES*-instance of the d-RDST problem and suppose that (D_1, X, k_1) is an instance of the annotated d-RDST problem reduced with respect to Rules 0 through 8 by repeatedly applying them on (D, k), by initially setting $X = \emptyset$. Then $|V(D_1)| \leq 8(k^2 + k)$.*

Proof. Since reduction rules map YES-instances to YES-instances and does not allow the parameter to increase, it is clear that (D_1, X, k_1) is a YES-instance of the annotated d-RDST problem and that $k_1 + |X| \leq k$. Therefore let T_1 be a solution out-branching of (D_1, X, k_1). A leaf of T_1 is either a vertex of out-degree zero in D_1 or a vertex of reduced degree. By Lemma 7.5, the total number of vertices of out-degree zero in D_1 is at most $k_1^2 \leq k^2$ and since T_1 is a solution out-branching, the total number of vertices of reduced degree is at most $k_1 + |X| \leq k$. Thus the number of leaves of T_1 is at most $k^2 + k$ and by Lemma 7.2 we have $|V(T_1)| \leq 4(k^2 + k + k^2 + k) = 8(k^2 + k)$. □

We now show how to obtain a kernel for the original (unannotated) version of the problem. Let (D, k) be an instance of the d-RDST problem and let (D', X, k') be the instance obtained by applying reduction rules 0 through 8 on (D, k) until no longer possible, by initially setting $X = \emptyset$. By Lemma 7.6, we know that if (D, k) is a YES-instance then $|V(D')| \leq 8(k^2 + k)$ and that $k' + |X| = k$. To get back an instance of the unannotated version, apply the following transformation on (D', X, k'). If $X \neq \emptyset$, add a directed path $Y = y_1, \ldots, y_{k+2}$ to D' and for $x \in X$ add the out-arc (x, y_i) for $1 \leq i \leq k + 2$. Call the resulting digraph D''.

We claim that (D', X, k') has a solution out-branching T' with at most $|X| + k'$ vertices of reduced degree and where all vertices in X have their degree reduced if and only if D'' admits an out-branching with at most k vertices of reduced degree. Let T' be a solution out-branching for (D', X, k'). To obtain a solution out-branching T'' of D'' simply add the path Y to T' and the out-arc (x_1, y_1). Clearly T'' has at most $k' + |X|$ vertices of reduced degree. Conversely let T'' be a solution out-branching for D''. First note that every vertex in X must be of reduced degree in T''. For if $x \in X$ is of full degree then $k + 1$ vertices $y_1, \ldots, y_{k+1}$ are of reduced degree, contradicting the fact that T'' has at most k vertices of reduced degree. Therefore we may assume that, in T'', vertex x_1 has an out-arc to the start vertex y_1 of the path $y_1, \ldots, y_{k+2}$ which appears as is in the out-branching. That is, we may assume that the vertices in the path Y are always of full degree in any out-branching and that there exists at most k vertices in $V(D'') \setminus (V(Y) \cup X)$ of reduced degree in T''. To obtain a solution out-branching T' for (D', X, k') simply delete the path Y from T''.

Since we add at most $k + 2$ vertices in this transformation, we have

Theorem 7.3. *The d-RDST problem, parameterized by the number of vertices of reduced degree, admits a problem kernel with at most $8k^2 + 9k + 2$ vertices.*

7.5 An Algorithm for the d-RDST Problem

In this section we describe a branching algorithm for the d-RDST problem with running time $O(5.942^k \cdot n^{O(1)})$. We first observe that in order to construct a solution out-branching of a given digraph, it is sufficient to know which vertices will be of reduced degree.

Lemma 7.7. *Let $D = (V, A)$ be a digraph and let X be the set of vertices of reduced degree in some out-branching of D. Given D and X, one can in polynomial time construct an out-branching of D in which the vertices of reduced degree is a subset of X.*

Proof. We describe an algorithm that constructs such an out-branching of D. Given D and X, our algorithm first constructs a digraph D' with vertex set $V(D)$ in which

1. all vertices in $V(D) \setminus X$ are connected to their out-neighbors in D by *solid* arcs;
2. a vertex $x \in X$ has a *dotted* out-arc to a vertex y if $(x, y) \in A(D)$ and y has no solid in-arc in D'.

We are guaranteed that there exists an out-branching of D' in which all solid arcs are present but in which one or more dotted arcs may be missing. Note that in D', a vertex with a solid in-arc has no other (solid or dotted) in-arcs.

Our algorithm now runs through all possible choices of the root of the proposed out-branching. For each choice of root, it does a modified breadth-first search (BFS) starting at the root. In the modified BFS-routine, when the algorithm visits a vertex v, it solidifies all dotted out-arcs from v, if any. For each dotted arc (v, w) that it solidifies, it deletes all dotted in-arcs to w. The algorithm then inserts the out-neighbors of v in the BFS-queue. If the BFS-tree thus constructed includes all vertices of D', the algorithm outputs this out-branching, or else, moves on to the next choice of root.

Claim. Suppose that r is the root of an out-branching of D' in which X is the vertex-set of reduced degree. Then the above algorithm, on selecting r as root, succeeds in constructing an out-branching in which the vertices of reduced degree is a subset of X.

In order to prove this claim, it is sufficient to show that in the BFS-tree T constructed by the algorithm with r as root, every vertex of D' is reachable from r. This suffices because every vertex in $V(D') \setminus X$ is of full degree in T.

Therefore let v be a vertex not reachable from r such that the distance, in D', from v to r is the shortest among all vertices not reachable from r in T. Let $r, v_1, \ldots, v_l, v$ be a shortest dipath from r to v in D'. By our choice of v, all vertices $v_1, \ldots, v_l$ are reachable from r in T. Note that the

arc (v_l, v) must have been dotted and in fact all in-arcs to v were dotted in D'. When the algorithm visited v_l, the only reason it could not solidify the arc (v_l, v) must have been because v *already* had a solid in-arc into it and hence the arc (v_l, v) had already been deleted. Suppose that v has a solid in-arc from u. Then u must have already been visited *before* v_l at which time the dotted arc (u, v) was solidified. But this means that u, and hence v, is reachable from r in the BFS-tree T, a contradiction. □

We now have an $O(k^{O(k)} \cdot n^{O(1)})$ algorithm for the d-RDST problem: Given an instance (D, k), we first obtain a kernel of size $O(k^2)$ using Theorem 7.3 and then run over all possible vertex-subsets X of the kernel of size at most k to determine the set of vertices of reduced degree. Then using Lemma 7.7, we verify whether one can indeed construct an out-branching in which the set of vertices of reduced degree is X.

RDST (D, X, k)
Input: A digraph $D = (V, A)$; $X \subseteq V$, such that the vertices in X will be of reduced degree in the out-branching that is being constructed; an integer parameter k. The algorithm is initially called after setting $X = \emptyset$.
Output: An out-branching of D in which every vertex of X is of reduced degree and with at most k vertices of reduced degree in total, if one exists, or NO, signifying that no such out-branching exists.

1. If $k < 0$ or $|X| > k$ return NO.
2. If no two vertices in $V(D) \setminus X$ have a common out-neighbor then
   (a) Reduce (D, X, k) with respect to Rules $1'$ through $5'$.
   (b) For each $(k - |X|)$-sized subset Y of $V(D) \setminus X$, check if there exists an out-branching of D in which the vertex set of reduced degree is $X \cup Y$. If yes, then "expand" this out-branching to an out-branching for the original instance and return the solution; else return NO.
3. Let $u, v \in V(D) \setminus X$ be two vertices with a common out-neighbor then
   (a) $X \leftarrow X \cup \{u\}$; $Z =$ Call **RDST**$(D, X, k-1)$.
   (b) If $Z \neq$ NO then return Z.
   (c) $X \leftarrow X \cup \{v\}$; Return **RDST**$(D, X, k-1)$.

Figure 7.3: Algorithm **RDST**.

In the rest of this section, we give an improved algorithm with running time $O(c^k \cdot n^{O(1)})$, for a constant c. Our algorithm (see Figure 7.3) is based on the simple observation that if two vertices u and v of the input digraph D have a common out-neighbor then one of them must be of reduced degree in *any* out-branching of D. The algorithm recurses on vertex-pairs that have a

common out-neighbor and, along each branch of the recursion tree, builds a set X of vertices which would be the candidate vertices of reduced degree in the out-branching that it attempts to construct. When there are no vertices to branch on, it reduces the instance (D, X, k) with respect to the following rules.

Rule 1′. If $u \in X$ and $d^+(u) = 1$, delete the out-arc from u and return (D, X, k). This is Rule 0 from Section 7.4.2.

Rule 2′. Let $u \in V(D)$ be of out-degree zero and let $v_1, \ldots, v_r$ be its in-neighbors. If $v_i \in X$ for all $1 \leq i \leq r$, assign v_1 as the parent of u and delete u. If there exists $1 \leq i \leq r$ such that $v_i \notin X$ then assign v_i as the parent of u and delete u. Return (D, X, k).

Rule 3′. This is Rule 8 from Section 7.4.2.

Rule 1′ is a reduction rule because a vertex of out-degree exactly one that is of reduced degree must necessarily lose its only out-arc. As for Rule 2′, we know that in the instance (D, X, k) obtained after the algorithm finishes branching, no two vertices of $V(D) \setminus X$ have a common out-neighbor and therefore at least $r - 1$ in-neighbors of u must be in X (and of reduced degree). If all in-neighbors of u are of reduced degree, we arbitrarily fix one of them as parent of u (so that we can construct an out-branching of the original instance later on) and delete u. If exactly $r-1$ in-neighbors of u are already of reduced degree, we choose that in-neighbor not in X as the parent of u and delete u. Also note that when applying Rule 3′ to a path $x_0, x_1, \ldots, x_{p-1}, x_p$, the vertices $x_0, x_1, \ldots, x_{p-1}$ are not in X, by Rule 1′. Therefore if Y is the set of in-neighbors of $x_1, \ldots, x_{p-1}$, excluding $\{x_0, x_1, \ldots, x_{p-2}\}$, then $Y \subseteq X$.

Observe the following:

1. By Rule 2′, no vertex in the reduced instance (D, X, k) has out-degree zero.

2. Every vertex in the subdigraph induced by $V(D) \setminus X$ has in-degree at most one and hence each connectivity component (a connected component in the undirected sense) is either a dicycle, or an out-tree or a dicycle which has out-trees rooted at its vertices. Thus each connectivity component has at most one dicycle and if a component does have a dicycle then it can be transformed into an out-branching by deleting an arc from the cycle. Such a digraph is called a *pseudo out-forest* [118].

We now reduce the instance (D, X, k) with respect to the following two rules:

Rule 4′. If at least $k+1-|X|$ connectivity components of $D[V \setminus X]$ contain dicycles, then return NO; else return (D, X, k).

Rule 5′. If a connectivity component of $D[V \setminus X]$ is a dicycle C such that no vertex in $V(C)$ has an out-neighbor in X, pick a vertex $u \in X$ with an arc to C and fix it as the "entry point" to C; delete C and set $k \leftarrow k-1$; return (D, X, k).

Rule 4′ is a reduction rule as every connectivity component that has a dicycle contains at least one vertex that will be of reduced degree. If the number of such components is at least $k+1-|X|$, one cannot construct an out-branching with at most k vertices of reduced degree where all vertices in X have their degree reduced. To see that Rule 5′ is a reduction rule, first note that since C has no out-arcs, it cannot contain the root of the proposed out-branching. Any path from the root to C must necessarily include a vertex from X and it does not matter which arc out of X we use to get to C, since every vertex in X has its degree reduced anyway. Moreover, in any out-branching, exactly one vertex of C must be of reduced degree. Therefore if (D', X', k') is the instance obtained by one application of Rule 5′ to the instance (D, X, k), then it is easy to see that these instances must be equivalent. Also note that each application of Rule 1′ through 5′ takes time $O(n+m)$.

Lemma 7.8. *Let (D, X, k) be an instance of the d-RDST problem in which no two vertices of $V(D) \setminus X$ have a common out-neighbor, and reduced with respect to Rules 1′ through 5′. Then $|V(D) \setminus X| \leq 7|X|$.*

Proof. Let D' be a digraph obtained from D by deleting all out-arcs from the vertices in X. Therefore in D', every vertex of X has out-degree zero and in-degree at most one. We show that a connectivity component of D' that has p vertices of X has at most $7p$ vertices of $V(D') \setminus X$. This will prove the lemma.

If a connectivity component of D' is an out-tree T', then every leaf of this out-tree is a vertex of X. If T' has p leaves, then applying Lemma 7.2 to T', we have that $|V(T')| \leq 8p$. Since exactly p of these vertices are from X, the number of vertices of $V(D') \setminus X$ in the out-tree is at most $7p$. Therefore let R be a connectivity component of D' containing a dicycle such that $|V(R) \cap X| = p$. Then R has exactly one dicycle, say C. By Rule 5′, C has a vertex x with an out-neighbor in $V(R) \setminus V(C)$, and therefore $d^+_R(x) \geq 2$. Let y be the out-neighbor of x in C. Delete the arc (x, y) to obtain an out-branching T with root y. Note that the number of leaves in T is the same as that in R. Moreover in transforming R to T, only one vertex (namely x) loses its out-degree. By Lemma 7.2,

$$|V(T)| \leq 4|x \cup V_0(T) \cup V_{\geq 2}(T)| \leq 4|V_0(T)| + 4|x \cup V_{\geq 2}(T)|,$$

and since $|x \cup V_{\geq 2}(T)| \leq 1 + (|V_0(T)| - 1) = p$, we have $|V(T)| \leq 8p$. Consequently $|V(R) \setminus X| \leq 7p$. This completes the proof of the lemma. □

To construct an out-branching, it is sufficient to choose the remaining $k-|X|$ vertices of reduced degree from the vertices in $V(D)\backslash X$. Setting $|X| = c$, the exponential term in the running time of the algorithm is bounded above by the function

$$\sum_{c=0}^{k} 2^c \cdot \binom{7c}{k-c} \leq k \cdot \max_{0 \leq c \leq k} 2^c \cdot \binom{7c}{k-c}.$$

We will show that the function $h(k) := \max_{0 \leq c \leq k} 2^c \cdot \binom{7c}{k-c}$ is bounded above by $(k+1) \cdot 5.942^k$.

Theorem 7.4. *Given a digraph D and a nonnegative integer k, one can decide whether D has an out-branching with at most k vertices of reduced degree, and if so, construct such an out-branching in time $O(5.942^k \cdot k^2 \cdot (n+m))$.*

Finally we prove the claimed upper-bound for the function $h(k)$. We first need a lemma.

Lemma 7.9. *For nonnegative integers $k \leq n$ and any real $x > 0$,*

$$\binom{n}{k} \leq \frac{(1+x)^n}{x^k}.$$

Proof. Since

$$x^{-k}(1+x)^k = x^{-k} \cdot \sum_{i=0}^{n} \binom{n}{i} x^i = \binom{n}{k} + \Delta,$$

where $\Delta = \sum_{0 \leq i \neq k \leq n} \binom{n}{i} x^{i-k} \geq 0$, the result follows immediately. □

Lemma 7.10. *For a nonnegative integer k, the function*

$$h(k) = \max_{0 \leq c \leq k} 2^c \cdot \binom{7c}{k-c}$$

is bounded above by $(k+1) \cdot 5.942^k$.

Proof. Using the bound in Lemma 7.9, we have that for any $0 \leq c \leq k$ and any $x > 0$,

$$2^c \cdot \binom{7c}{k-c} \leq 2^c \cdot \frac{(1+x)^{7c}}{x^{k-c}} = \frac{\left(2x(1+x)^7\right)^c}{x^k}.$$

Since the above inequality holds for *any* $x > 0$, it holds, in particular, for the positive roots of the equation $2x(1+x)^7 - 1 = 0$. This equation has exactly one positive root and its value is approximately 0.16830. Substituting this value in the previous inequality, we obtain

$$2^c \cdot \binom{7c}{k-c} \leq \frac{1}{0.16830^k} \leq 5.942^k,$$

and since there are $k+1$ such terms in $h(k)$, the value of $h(k)$ is at most $(k+1) \cdot 5.942^k$. □

7.6 Concluding Remarks

We studied a natural generalization of the Full Degree Spanning Tree problem to directed graphs. We showed that the d-FDST problem is W[1]-hard even on the class of DAGs and that the d-RDST problem is fixed-parameter tractable. For the d-RDST problem, we obtained a kernel with at most $8k^2 + 9k + 2$ vertices and an algorithm with running time $O(5.942^k \cdot (n+m))$. Natural open questions are to investigate whether d-RDST admits a linear-vertex kernel and design algorithms with better running times.

Chapter 8

Summary and Future Research

In this thesis, we studied different parameterizations of NP-optimization problems with the objective of identifying parameterizations that are most likely to be useful in practice.

We started out in Chapter 2 with NP-optimization problems that are fixed-parameter tractable to begin with. The parameter in such cases always assumes a value that is a significant fraction of the input size and we argued that such a parameterization is not useful in practice since any FPT-algorithm for the problem runs in time that is essentially exponential in the input-size. For such problems the natural parameter is the difference between the optimum and the guaranteed lower or upper-bound. In Chapter 3 we tried to extend these ideas to the realm of approximation algorithms.

From Chapters 4 through 7, we saw several situations where an NP-optimization problem is parameterizable in more than one way. If the problem at hand is W-hard for one parameter, we tried a different parameterization for which it is in FPT. Such a strategy gives us an idea as to the best way to tackle the intractability of a problem. In this final chapter we summarize the results presented in this thesis and collate the open problems that appear throughout this thesis.

8.1 Parameterizing Problems Beyond Default Values

We showed that there exist many NP-optimization problems with the property that their optimum solution size is bounded below by an unbounded function of the input size. The standard parameterized version of these problems is not interesting as they are trivially in FPT. We argued that the natural way to parameterize such problems is to let the parameter represent the deficit between the optimum and the lower bound, that is, to parameterize above the guaranteed lower bound. But we also observed that the lower bound must be tight in some sense in order for the above-guarantee parameterizations to be interesting. This led us to introduce tight lower and

upper bounds. We showed that parameterizations of the type $\text{TLB}+\epsilon\cdot|I|+k$ or $\text{TUB}-\epsilon\cdot|I|-k$, that are sufficiently beyond tight bounds, are not in FPT, unless P = NP.

There are quite a few open problems. Firstly we would like to have general results about classes of NP-optimization problems whose optima are bounded below (above) by a nontrivial lower (upper) bound.

Open Problem 8.1. Is there a characterization for the class of problems for which the above or below-guarantee question with respect to a tight lower or upper bound is in FPT (or W[1]-hard)?

There are several directions to pursue from an algorithmic point of view. Not many results are known on parameterized above or below-guarantee problems and, in particular, the complexity of problems (1) through (7) in Chapter 2 Section 2.4, when parameterized above their guaranteed values, is open. Some of the more interesting above-guarantee problems are:

Open Problem 8.2. Planar Independent Set: Given an n-vertex planar graph and an integer parameter k, does G have an independent set of size at least $\lceil n/4\rceil + k$?

Open Problem 8.3. Max Exact c-Sat: Given a Boolean CNF formula F with m clauses such that each clause has exactly c distinct literals and an integer parameter k, does there exist an assignment that satisfies at least $(1-2^{-c})m + k$ clauses

Recently Gutin et al. [74] have shown that Max Lin-2, Max Acyclic Subgraph and a special case of Max Exact c-Sat are in FPT. In an another paper [75], the same authors show that Max 2-Sat is in FPT when parameterized above the bound $3m/4$ by demonstrating a kernel with $O(k^2)$ variables. Finally in 2010, Alon et al. [4] showed that Max Exact c-Sat is in FPT and that it admits a kernel with $O(k^2)$ variables.

Here is an interesting below-guarantee problem whose parameterized complexity is open. Recall that an undirected graph G is *perfect* if for every induced subgraph H of G, the chromatic number and clique number of H are equal.

Open Problem 8.4. [36, 105] Perfect Vertex Deletion: Let G be a graph on n vertices and m edges and k a nonnegative integer. Does there exist a vertex-induced subgraph on $n-k$ vertices that is perfect? A similar question can be framed for the edge version. What is the approximability of this problem?

8.2 Kőnig Subgraph Problems

We studied a set of subgraph problems, called Kőnig Subgraph problems, which are defined as follows:

1. Kőnig Vertex (Edge) Deletion (KVD/KED). Given a graph G and a nonnegative integer k, does there exist at most k vertices (respectively, edges) whose deletion results in a Kőnig subgraph?

2. Vertex (Edge) Induced Kőnig Subgraph (VKS/EKS). Given a graph G and a nonnegative integer k, does there exist at least k vertices (respectively, edges) which induce a Kőnig subgraph?

We showed that KVD is in FPT whereas VKS is W[1]-hard. The EKS problem is in FPT and we conjecture that KED is W[1]-hard.

Open Problem 8.5. What is the parameterized complexity of the Kőnig Edge Deletion problem?

Our algorithm for KVD uses the FPT-algorithm for Above Guarantee Vertex Cover as a subroutine, which in turn, uses the one for Min 2-Cnf Del developed by Razgon and O'Sullivan [116].

Open Problem 8.6. Design FPT-algorithms for Kőnig Vertex Deletion and Above Guarantee Vertex Cover with a run-time better than $O(15^k \cdot k^2 \cdot |E|^3)$, perhaps without using the algorithm for Min 2-Cnf Del.

With regards to polynomial-time approximability, we show that if the Unique Games Conjecture is true, the problems KVD, KED, and Min 2-Cnf Del do not admit constant-factor approximation algorithms; VKS is as hard to approximate as Independent Set; and EKS admits a factor-5/3 algorithm.

We also showed that any graph $G = (V, E)$ with maximum matching M has an edge-induced Kőnig subgraph of size at least

$$\frac{3|E_M|}{4} + \frac{|E - E_M|}{2} + \frac{|M|}{4},$$

where E_M is the set of edges in the graph $G[V(M)]$.

Open Problem 8.7. Is this lower bound tight? What is the parameterized complexity of the following above-guarantee Edge Induced Kőnig Subgraph problem: Given a graph $G = (V, E)$ with maximum matching M and an integer parameter k, does G have an edge-induced Kőnig subgraph of size at least

$$\frac{3|E_M|}{4} + \frac{|E - E_M|}{2} + \frac{|M|}{4} + k?$$

8.3 Unique Coverage

We considered several plausible parameterizations of the Unique Coverage problem and showed that except for the standard parameterized version (and a generalization of it), all other versions are unlikely to be fixed-parameter intractable. We also described problem kernels for several special

cases of the standard parameterized version of Unique Coverage and showed that the general version admits a kernel with at most 4^k sets. We noted that this is essentially the best possible kernel due to a lower bound result by Dom et al. [46].

We gave FPT-algorithms for Budgeted Unique Coverage and several of its variants using the well-known method of color-coding. Our randomized algorithms have good running times but the deterministic algorithms make use of perfect hash families and this introduces large constants in the running times, a common enough phenomenon when derandomizing randomized algorithms using such function families [6]. It would be interesting to obtain faster algorithms.

Open Problem 8.8. Is there an FPT-algorithm for Unique Coverage with run-time $O^*(2^{O(k)})$? For Budgeted Unique Coverage, is there an algorithm with run-time $O^*(2^{O(k \log\log B)})$?

For derandomizing our algorithms, we used (k, s)-hash families. We also showed the existence of such families of size $(2e)^{sk/t} \cdot s \log n$, which is smaller than that of the best-known (n, t, s)-perfect hash families.

Open Problem 8.9. Give an explicit construction of an (k, s)-hash family with domain $[n]$, range $[t]$ and of size $O(2^{O(sk/t)} \cdot s \log n)$.

We also showed that given an instance $(\mathcal{U}, \mathcal{F}, k)$ of the Unique Coverage problem, one can always cover at least $|\mathcal{U}|/8 \log M$ elements, where $M = \max_{S \in \mathcal{F}} |S|$.

Open Problem 8.10. Is this lower bound tight? What is the parameterized complexity of the above-guarantee Unique Coverage problem: given an instance $(\mathcal{U}, \mathcal{F}, k)$, does there exist a subfamily $\mathcal{F}'$ of $\mathcal{F}$ that uniquely covers at least $|\mathcal{U}|/8 \log M + k$ elements, where $M = \max_{S \in \mathcal{F}} |S|$?

8.4 NP-Optimization Problems on Directed Graphs

We considered two problems on directed graphs. The first problem was that of finding an induced subdigraph with k vertices in a given digraph that satisfied some fixed hereditary property. We completely characterized hereditary properties for which this problem is in FPT or W[1]-complete. We first did this for general directed graphs and then for oriented graphs. We also characterized hereditary properties for which the induced subgraph problem is W[1]-complete in general directed graphs but in FPT for oriented graphs.

In this context, we consider a related problem called the Graph Modification problem $\mathcal{P}(i, j, k)$ which asks whether a given input graph G can be 'modified' by deleting at most i vertices, j edges and adding at

most k edges so that the resulting graph satisfies property $\mathcal{P}$. Cai [25] has shown that if a hereditary property $\mathcal{P}$ has a finite forbidden set, the graph modification problem $\mathcal{P}(i,j,k)$ is fixed-parameter tractable with parameters i,j,k. There is no general result for the case when the forbidden set is infinite. For instance, the Odd Cycle Transversal problem is fixed-parameter tractable [117] whereas the Wheel-Free Vertex (Edge) Deletion problem is W[2]-hard [95].

Open Problem 8.11. Is there a characterization of hereditary properties with an infinite forbidden set for which the Graph Modification problem is either in FPT (or W-hard)?

The other problem on directed graphs that we studied is a generalization of the (undirected) Full-Degree Spanning Tree problem. We showed that the d-FDST problem is W[1]-hard even on the class of DAGs and that the d-RDST problem is fixed-parameter tractable. For the d-RDST problem, we obtained a kernel with at most $8k^2 + 9k + 2$ vertices and an algorithm with run-time $O(5.942^k \cdot n^{O(1)})$.

Open Problem 8.12. Is there a sub-quadratic kernel for d-RDST? Design an FPT-algorithm for d-RDST with a run-time better than $O(5.942^k \cdot n^{O(1)})$.

8.5 Concluding Remarks

Parameterized Complexity has emerged as an important discipline both from a theoretical and a practical perspective. The study of fixed-parameter algorithms and kernelization, in particular, has led to new insights into the hardness of NP-complete problems and has fostered the development of interesting algorithmic techniques that have proved useful in practice. In this thesis we tried to address issues relating to feasible parameterizations of problems that may be useful in practice. As we have seen in this chapter, there are many open questions. With the rapid development that this subject is witnessing, we can expect at least some of them to be resolved satisfactorily in the near future.

Publications

1. M. Mahajan, V. Raman and S. Sikdar. *Parameterizing* NP*-optimization Problems Above or Below Guaranteed Values.* Journal of Computer and System Sciences, Volume 75, Issue 2, Pages 137-153, 2009. A preliminary version appeared in Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006), Springer-Verlag, LNCS Volume 4169, Pages 38-49, 2006.
2. V. Raman and S. Sikdar. *Approximating Beyond the Limit of Approximation.* Manuscript.
3. S. Mishra, V. Raman, S. Saurabh, S. Sikdar and C.R. Subramanian. *The Complexity of Kőnig Subgraph Problems and Above-Guarantee Vertex Cover.* To appear in Algorithmica. This paper was an amalgamation of the following two papers.
   (a) S. Mishra, V. Raman, S. Saurabh, S. Sikdar and C. R. Subramanian. *The Complexity of Finding Subgraphs whose Matching Number Equals the Vertex Cover Number.* In Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC 2007), Springer LNCS Volume 4835, Pages 268-279, 2007.
   (b) S. Mishra, V. Raman, S. Saurabh and S. Sikdar. *Kőnig Deletion Sets and Vertex Covers Above the Matching Size.* In Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC 2008), Springer LNCS Volume 5369, Pages 836-847, 2008.
4. H. Moser, V. Raman and S. Sikdar. *The Complexity of the Unique Coverage Problem.* In Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC 2007), Springer LNCS Volume 4835, Pages 621-631, 2007.
5. N. Misra, V. Raman, S. Saurabh and S. Sikdar. *Budgeted Unique Coverage and Color-Coding.* In Proceedings of the 4th Computer Science Symposium in Russia, CSR 2009, Springer LNCS, Volume 5675, Pages 310-321, 2009.
6. V. Raman and S. Sikdar. *Parameterized Complexity of the Induced Subgraph Problem in Directed Graphs.* Information Processing Letters, Volume 104, Pages 79-85, 2007.
7. D. Lokshtanov, V. Raman, S. Saurabh and S. Sikdar. *On the Directed Degree Preserving Spanning Tree Problem.* In Proceedings of the Fourth International Workshop on Parameterized and Exact Computation (IWPEC 2009), Springer LNCS, Volume 5917, Pages 276-287, 2009.

Bibliography

[1] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and Directed Cut problems. In *Proceedings of the 37th annual ACM symposium on Theory of computing (STOC 2005)*, pages 573–581. ACM, 2005.

[2] N. Alon, G. Cohen, M. Krivelevich, and S. Litsyn. Generalized hashing and parent-identifying codes. *Journal of Combinatorial Theory Series A*, 104(1):207–215, 2003.

[3] N. Alon, F. V. Fomin, G. Gutin, M. Krivelevich, and S. Saurabh. Spanning directed trees with many leaves. *SIAM Journal of Discrete Math*, 2009. To appear.

[4] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX r-SAT above a tight lower bound. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA10)*. SIAM, 2010.

[5] N. Alon and J. H. Spencer. *The Probabilistic Method*. Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 2000.

[6] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

[7] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchett-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[10] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the Undirected Feedback Vertex Set problem. *SIAM Journal Discrete Mathematics*, 12(3):289–297, 1999.

[11] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[12] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Monographs in Mathematics. Springer, 2000.

[13] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45:104–126, 1992.

[14] A. Barg, G. Cohen, S. Encheva, G. Kabatiansky, and G. Zémor. A hypergraph approach to the identifying parent property: The case of multiple parents. *SIAM Journal on Discrete Mathematics*, 14(3):423–431, 2001.

[15] N. Betzler, M. R. Fellows, J. Guo, R. Niedermeier, and F. A. Rosamond. Fixed-parameter algorithms for Kemeny Scores. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM 2008)*, volume 5034 of *LNCS*, pages 60–71. Springer, 2008.

[16] R. Bhatia, S. Khuller, R. Pless, and Y. Sussmann. The Full-Degree Spanning Tree problem. *Networks*, 36(4):203–209, 2000.

[17] A. Billionnet. On interval graphs and matrix profiles. *RAIRO Operations Research*, 20:245–256, 1986.

[18] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels (extended abstract). In *Proceedings of 35th International Colloquium of Automata, Languages and Programming (ICALP 2008)*, LNCS, pages 563–574. Springer, 2008.

[19] H. L. Bodlaender, M. R. Fellows, and M. T. Hallett. Beyond NP-completeness for problems of bounded width: Hardness for the W-hierarchy (extended abstract). In *ACM Symposium on Theory of Computing*, pages 449–458, 1994.

[20] H. L. Bodlaender and A. M. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3), 2007.

[21] N. Bourgeois, B. Escoffier, and V. Paschos. Efficient approximation by "low-complexity" exponential algorithms. Technical Report 271, LAMSADE, Université Paris-Dauphine, 2008. Submitted for publication.

[22] J. Bourjolly and W. Pulleyblank. König-Egerváry graphs, 2-bicritical graphs and fractional matchings. *Discrete Applied Mathematics*, 24:63–82, 1989.

[23] H. J. Broersma, A. Huck, T. Kloks, O. Koppios, D. Kratsch, H. Müller, and H. Tuinstra. Degree-preserving forests. *Networks*, 35(1):26–39, 2000.

[24] J. F. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[25] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[26] L. Cai and J. Chen. On fixed-parameter tractability and approximability of NP-optimization problems. *Journal of Computing and System Sciences*, 54(3):465–474, 1997.

[27] L. Cai and X. Huang. Fixed-parameter approximation: Conceptual framework and approximability results. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation*, volume 4169 of *LNCS*, pages 96–108. Springer, 2006.

[28] M. Cesati. Perfect Code is W[1]-complete. *Information Processing Letters*, 81(3):163–168, 2002.

[29] J. Chen and I. A. Kanj. On approximating minimum vertex cover for graphs with perfect matching. *Theoretical Computer Science*, 337(1-3):305–318, 2005.

[30] J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[31] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the Directed Feedback Vertex Set problem. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC 2008)*, pages 177–186. ACM, 2008.

[32] J. Chen, S. Lu, S.-H. Sze, and F. Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, pages 298–307, 2007.

[33] Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *LNCS*, pages 109–120. Springer, 2006.

[34] J. Cherian and R. Thurimella. Approximating minimum size k-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000.

[35] M. Chlebík and J. Chlebíková. Minimum 2SAT-Deletion: inapproximability results and relations to minimum vertex cover. *Discrete Applied Mathematics*, 155(2):172–179, 2007.

[36] M. Chudnovsky, G. Cornuéjols, X. Liu, P. D. Seymour, and K. Vušković. Recognizing Berge graphs. *Combinatorica*, 25(2):143–186, 2005.

[37] N. Cohen, F. V. Fomin, G. Gutin, E. Kim, S. Saurabh, and A. Yeo. Algorithms for finding k-Vertex Out-Trees and its applications to the k-Internal Out-Branching problem. In *Proceedings of the 15th Conference on Computing and Combinatorics (COCOON 2009)*, volume 5609 of *LNCS*, pages 37–46. Springer, 2009.

[38] T. H. Cormen, C. E. Lieserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. M.I.T Press, 2nd edition, 2001.

[39] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of graph grammars and computing by graph transformation: Volume I. Foundations*, pages 313–400. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.

[40] M. Cygan, Ł. Kowalik, M. Pilipczuk, and M. Wykurz. Exponential-time approximation via instance reduction. http://arxiv.org/abs/0810.4934, October 2008.

[41] J. Daligault, G. Gutin, E. Kim, and A. Yeo. FPT-algorithms and kernels for the Directed k-Leaf Problem. Manuscript.

[42] E. Dantsin, M. Gavrilovich, E. A. Hirsch, and B. Konev. Max Sat approximation: Beyond the limits of polynomial-time approximation. *Annals of Pure and Applied Logic*, 113(1-3):81–94, 2001.

[43] E. D. Demaine, M. T. Hajiaghayi, U. Feige, and M. R. Salavatipour. Combination can be hard: approximability of the Unique Coverage problem. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 162–171, 2006.

[44] R. Deming. Independence numbers of graphs—an extension of the König-Egerváry theorem. *Discrete Mathematics*, 27:22–33, 1979.

[45] I. Dinur and S. Safra. On the hardness of approximating Minimum Vertex Cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[46] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through Colors and IDs. In *Proceedings of 36th International Colloquium of Automata, Languages and Programming (ICALP 2009)*, volume 5555 of *LNCS*. Springer, 2009. To appear.

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

[48] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation algorithms. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *LNCS*, pages 121–129. Springer, 2006.

[49] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods revisited. Manuscript.

[50] T. N. Ephraim Korach and B. Peis. Subgraph characterization of red/blue-split graphs and König-Egerváry graphs. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, 2006.

[51] T. Erlebach and E. J. van Leeuwen. Approximating geometric coverage problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, pages 1267–1276, 2008.

[52] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A.Rosamond. FPT is P-time extremal structure-I. In *Proceedings of Algorithms and Complexity in Durham (ACiD 2005)*, 2005.

[53] G. Even, J. S. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

[54] S. S. Fedin and A. S.Kulikov. A $2^{|E|/4}$-time algorithm for Max Cut. *Journal of Mathematical Sciences*, 126(3):1200–1204, 2005.

[55] U. Feige. A threshold of $\ln n$ for approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998.

[56] U. Feige and J. Killian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

[57] M. R. Fellows. Personal communication. 2006.

[58] H. Fernau, F. V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernels: On out-trees with many leaves. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, pages 421–432, 2009.

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

[60] F. V. Fomin, S. Gaspers, and A. V. Pyatkin. Finding a Minimum Feedback Vertex Set in time $O(1.7548^n)$. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *LNCS*, pages 184–191. Springer, 2006.

[61] F. V. Fomin, S. Gaspers, S. Saurabh, and S. Thomasse. A linear vertex kernel for maximum internal spanning tree. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, volume 5878 of *LNCS*, pages 275–282. Springer, 2009.

[62] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 18–25, 2006.

[63] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 133–142. ACM, 2008.

[64] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1981.

[65] S. Gaspers, S. Saurabh, and A. Stepanov. A moderately exponential-time algorithm for Full-Degree Spanning Tree. In *Proceedings of the 5th Annual Conference on Theory and Applications of Models of Computation (TAMC 2008)*, volume 4978 of *LNCS*, pages 478–489. Springer, 2008.

[66] F. Gavril and M. Yannakakis. Edge dominating sets in graphs. *Journal of Applied Mathematics*, 38(3):364–372, 1980.

[67] M. X. Goemans. Minimum bounded degree spanning trees. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 273–282. IEEE Computer Society, 2006.

[68] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for Maximum Cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[69] R. L. Graham, B. L. Rothschild, and J. H. Spencer. *Ramsey Theory*. John Wiley & Sons, 1980.

[70] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression based fixed-parameter algorithms for Feedback Vertex Set and Edge Bipartization. *Journal of Computer and System Sciences*, 2006.

[71] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.

[72] J. Guo, R. Niedermeier, and S. Wernicke. Fixed-parameter tractability results for Full-Degree Spanning Tree and its dual. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *LNCS*, pages 203–214. Springer, 2006.

[73] G. Gutin, E. Kim, and I. Razgon. Minimum Leaf Out-Branching and related problems. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM 2008)*, volume 5034 of *LNCS*, pages 235–246. Springer, 2008.

[74] G. Gutin, E. Kim, S. Szeider, and A. Yeo. A probabilistic approach to problems parameterized above tight lower bound. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC 2009)*, LNCS. Springer, 2009. To appear.

[75] G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX-2-SAT above a tight lower bound. Manuscript available at: `http://eprintweb.org/S/article/cs/0907.4573`, 2009.

[76] G. Gutin, A. Rafiey, S. Szeider, and A. Yeo. The Linear Arrangement problem parameterized above-guaranteed value. In *Proceedings of the 6th Conference on Algorithms and Complexity (CIAC 2006)*, volume 3998 of *LNCS*, pages 356–367. Springer, 2006.

[77] G. Gutin, S. Szeider, and A. Yeo. Fixed-parameter complexity of Minimum Profile problems. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *LNCS*, pages 60–71. Springer, 2006.

[78] G. Gutin and A. Yeo. Some parameterized problems on digraphs. *The Computer Journal*, 51(3):363–371, 2008.

[79] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 5:285–306, 1965.

[80] J. Håstad. Clique is hard to approximate to within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[81] J. Håstad and S. Venkatesh. On the advantage over a random assignment. *Random Structures & Algorithms*, 25(2):117–149, 2004.

[82] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger. Interval completion with few edges. In *Proceedings of the 39th annual ACM symposium on Theory of computing (STOC 2007)*, pages 374–381. ACM, 2007.

[83] S. Jukna. *Extremal Combinatorics*. Springer-Verlag, Berlin, 2001.

[84] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1998.

[85] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC 2002)*, pages 767–775. ACM, 2002.

[86] S. Khot and V. Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289:997–1008, 2002.

[87] S. Khot and O. Regev. Vertex Cover might be hard to approximate to within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[88] S. Khuller, R. Bhatia, and R. Pless. On local search and placement of meters in networks. *SIAM Journal on Computing*, 32(3):470–487, 2003.

[89] S. Khuller, A. Moss, and J. Naor. The Budgeted Maximum Coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[90] P. N. Klein, S. A. Plotkin, S. Rao, and E. Tardos. Approximation algorithms for Steiner and Directed Multicuts. *Journal of Algorithms*, 22(2):241–269, 1997.

[91] T. Kloks. *Treewidth: Computations and Approximations*. Springer, 1994.

[92] J. Kneis and P. Rossmanith. A new satisfiability algorithm with applications to Max Cut. Technical Report AIB-2005-08, Department of Computer Science, RWTH Aachen, 2005.

[93] J. M. Lewis and M. Yannakakis. The Node-Deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[94] Y. Lin and J. Yuan. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica (English Series)*, 10(1):107–112, 1994.

[95] D. Lokshtanov. Wheel-Free Deletion is W[2]-hard. In *Proceedings of the 3rd International Workshop on Exact and Parameterized Computation (IWPEC 2008)*, volume 5018 of *LNCS*, pages 141–147. Springer, 2008.

[96] L. Lovász. Ear-decompositions of matching-covered graphs. *Combinatorica*, 3:105–118, 1983.

[97] L. Lovász and M. D. Plummer. *Matching Theory*. North Holland, 1986.

[98] M. Mahajan and V. Raman. Parameterizing above-guaranteed values: Max Sat and Max Cut. *Journal of Algorithms*, 31(2):335–354, 1999.

[99] D. Marx. Chordal Deletion is fixed-parameter tractable. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2006)*, volume 4217 of *LNCS*, pages 37–48. Springer, 2006.

[100] D. Marx and I. Schlotter. Obtaining a planar graph by vertex deletion. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2007)*, volume 4769 of *LNCS*, pages 292–303. Springer, 2007.

[101] K. Mehlhorn. On the program size of perfect and universal hash functions. In *Proceedings of the 23th IEEE Symposium on Foundations of Computer Science (FOCS 1982)*, pages 170–175. IEEE, 1982.

[102] S. Mishra, V. Raman, S. Saurabh, S. Sikdar, and C. R. Subramanian. The complexity of finding subgraphs whose matching number equals the vertex cover number. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC 2007)*, volume 4835 of *LNCS*, pages 268–279. Springer, 2007.

[103] H. Moser and D. M. Thilikos. Parameterized complexity of finding regular induced subgraphs. *Journal of Discrete Algorithms*, 2008.

[104] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[105] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge-modification problems. In *Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1999)*, volume 1665 of *LNCS*, pages 65–77. Springer, 1999.

[106] G. Nemhauser and L. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

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

[108] R. Niedermeier and P. Rossmanith. New upper bounds for Maximum Satisfiability. *Journal of Algorithms*, 36(1):63–88, 2000.

[109] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[110] K. Pietrzak. On the parameterized complexity of the fixed alphabet Shortest Common Supersequence and Longest Common Subsequence problems. *Journal of Computing and System Sciences*, 67(4):757–771, 2003.

[111] S. Poljak and D. Turzík. A polynomial algorithm for constructing a large bipartite subgraph with an application to a satisfiability problem. *Canadian Journal of Mathematics*, 34(3):519–524, 1982.

[112] I. Pothof and J. Schut. Graph-theoretic approach to identifiability in a water distribution network. Memorandum 1283, Universiteit Twente, Twente, The Netherlands, 1995.

[113] E. Prieto and C. Sloper. Reducing to independent set structure: The case of k-Internal Spanning Tree. *Nordic Journal of Computing*, 12(3):308–318, 2005.

[114] V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science*, 351(3):446–458, 2006.

[115] V. Raman, S. Saurabh, and C. R. Subramanian. Faster fixed-parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms*, 2(3):403–415, 2006.

[116] I. Razgon and B. O'Sullivan. Almost 2-Sat is fixed-parameter tractable (extended abstract). In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, pages 551–562. Springer-Verlag, 2008. To appear in the Journal of Computer and System Sciences.

[117] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[118] S. Saurabh. *Exact Algorithms for Optimization and Parameterized Versions of Some Graph-Theoretic Problems*. PhD thesis, Homi Bhaba National Institute, Mumbai, India, August 2007.

[119] J. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal on Computing*, 19(5):775–786, 1990.

[120] H. Schröder, A. E. May, O. Sýkora, and I. Vrto. Approximation algorithms for the Vertex Bipartization problem. In *Proceedings of SOFSEM '97*, volume 1338 of *LNCS*, pages 547–554. Springer, 1997.

[121] F. Sterboul. A characterization of graphs in which the transversal number equals the matching number. *Journal of Combinatorial Theory, Series B*, 27:228–229, 1979.

[122] V. Vassilevska, R. Williams, and S. L. M. Woo. Confronting hardness using a hybrid approach. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm (SODA 2006)*, pages 1–10, New York, NY, USA, 2006. ACM.

[123] V. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.

[124] S. Vishwanathan. On hard instances of approximate vertex cover. *ACM Transactions on Algorithms*, 5(1):1–6, 2008.

[125] D. West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001.

[126] R. Williams. A new algorithm for optimal 2-Constraint Satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[127] D. Zuckerman. Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. In *Proceedings of the 38th ACM Symposium on Theory of Computing (STOC 2006)*, pages 681–690, 2006.