Algorithms for Parameterized Constraint Satisfaction Problems

Robert Edward Crowston

A thesis submitted to Royal Holloway, University of London
for the degree of Doctor of Philosophy

May 2013

Declaration

I Robert Edward Crowston hereby declare that this thesis is the record of work carried out by me. Wherever contributions of others are involved, this is indicated clearly, with due reference to the literature and acknowledgement of collaborative research and discussions.

Abstract

There are many problems requiring superpolynomial time in the size of the instance. Parameterized complexity considers the problem having an additional parameter, isolating the superpolynomial behaviour. A problem is fixed-parameter tractable if it has an algorithm which has running time polynomial in the size of the instance multiplied by a (usually superpolynomial) function of the parameter.

This thesis considers the parameterized complexity of several problems where the aim is to satisfy a set of constraints. For such problems, it is known that a certain proportion of constraints can be satisfied. We consider the parameterized complexity of such problems with respect to the number of constraints satisfied above this, as well as with respect to other structural parameters.

Acknowledgements

I would like to thank my supervisors, Gregory Gutin and Anders Yeo for their support and encouragement. In particular, thanks go to Gregory for sharing his wisdom and being generous with his time. I would also like to thank my examiners, Mark Jerrum and Stefan Szeider for providing thoughtful feedback and discussion.

I would like to thank my co-authors: Michael Fellows, Mark Jones, Eun Jung Kim, Gabriele Muciaccia, Venkatesh Raman, Frances Rosamond, Imre Z. Ruzsa, Saket Saurabh and Stéphan Thomassé. Their encouragement and enthusiasm have made my studies all the more enjoyable.

Contents

Chapter 1

Introduction

In computational complexity, it is desirable to obtain an efficient algorithm to solve a given problem. Ideally, one would like to obtain a polynomial-time algorithm, that is show the problem is in the class P. However many problems are unlikely to have a polynomial-time algorithm. Such problems have been studied in order to obtain practically useful solutions efficiently. For example, in the study of approximation algorithms one relaxes the aim of finding an optimal solution, and instead searches for a solution within a certain ratio of the optimal solution.

One might instead sacrifice a guaranteed running time, and instead produce an algorithm that is often efficient, but in the worst case may be very slow. What if, in fact, the algorithm appeared to run fast in practical instances, even if the theoretical running time was poor? This might suggest that practical instances have extra structural properties, and in which case, the theoretical study of how a problem behaves under such restrictions would be highly worthwhile.

Downey and Fellows [33] initialized the study of parameterized complexity. Given a problem x, and a *parameter* k, one is interested in obtaining an algorithm with running time $f(k) \cdot n^{O(1)}$. By doing so, we contain the combinatorial explosion to the parameter k. When k is small, it is often the case that $f(k)$ is not too large, so the algorithm has a practical running time.

We have a choice as to what meaning is applied to the parameter. In the simplest form, given a maximization problem, the parameterized form of the problem might ask for a solution of size/value at least k (for a minimization problem, one might instead ask for a solution of size at most k). As we will see later on, it might also be interesting to consider the question of finding a solution of size at least $g(x) + k$, where g is a problem-specific function we will introduce later.

One might also use k to capture structural properties. For example, if the problem was on a graph, one might restrict the treewidth of the graph to at most k. Of course, looking at how structural properties influence the running time of algorithms for a problem has been considered before the framework of parameterized complexity was introduced. However, the framework has opened the field up to a much more systematic study.

It is often the case that a fixed-parameter tractable (FPT) algorithm for a problem can be

demonstrated by applying some polynomial-time reductions to the problem, obtaining an equivalent instance with size bounded by a function of k. Using any known algorithm on the newly created instance will thus solve the problem in FPT time. Such a reduction is called a *kernelization*. The study of the existence, or lack of existence, of polynomial-sized kernels has received considerable interest in recent years.

We will focus on the parameterizations of constraint satisfaction problems. In such problems, one is given a set of constraints, and a set of variables, with the aim of finding an assignment to the variables to satisfy all the constraints. In the maximization version of such problems, one might wish to obtain an assignment maximizing the number of constraints satisfied. For each problem, we will explore the possible natural parameterizations, whether there exists an FPT algorithm for the problem, and whether the problem has a kernel. We will also show that some parameterizations do not have an FPT algorithm, assuming reasonable complexity-theory assumptions.

A related notion to that of FPT is that of kernelization. Informally, a *kernelization algorithm* is an algorithm that maps every instance to a new instance of size at most $f(k)$, such that the new instance is a YES-instance if and only if the original instance is a YES-instance (see Section 2.3 for the formal definition of a parameterized problems and kernelization). The *size* of the kernel is $f(k)$.

Kernelization algorithms are advantageous in several ways. Since the kernelization algorithm occurs as an efficient pre-processing stage, it may be studied independently of the algorithm to solve the reduced instance. We are often concerned with finding polynomial-sized kernels, and if the kernel is polynomial, minimizing the degree of the polynomial. There are also tools to enable us to show the non-existence of polynomial kernels, or non-existence of a polynomial kernel of a specific size, under reasonable complexity-theory assumptions (more precisely, unless coNP$\subseteq$NP/poly [9, 11, 32]). Thus, for a particular problem it is possible to show the existence of a kernel, and that the kernel is of optimal size.

If a problem has a kernel, then it is fixed-parameter tractable, by observing that firstly applying a kernelization algorithm, then applying any known algorithm to the reduced instance gives a fixed-parameter tractable algorithm. In fact, the other direction of this implication holds too, as the following well-known lemma shows.

Lemma 1. *A parameterized problem is fixed-parameter tractable if and only if it is decidable and has a kernel.*

Proof. Given an algorithm for the problem with running time $h(|x|)$ and a kernelization with kernel of size $g(k)$, by firstly running the kernelization algorithm, and then solving the resulting instance in time $(h(g(k))$ a fixed-parameter tractable algorithm is formed.

In the other direction, assume there is a fixed parameter tractable algorithm with running time $f(k) \cdot |x|^c$ for some constant c. Run this algorithm for time $|x|^{c+1}$. If this algorithm terminates, the reduced instance consists of a trivial YES or NO-instance of the problem. If the algorithm does not terminate, then $|x|^{c+1} \leq |x|^c \cdot f(k)$. So $|x| \leq f(k)$, and the instance is already a kernel of size

$f(k)$. □

Note that the kernel generated by this proof is likely to be large.

We now consider the Vertex Cover problem, as an example of the use of FPT and kernelizations.

Vertex Cover
Instance: A graph $G = (V, E)$ and a nonnegative integer k.
Parameter: The integer k.
Question: Does there exist a vertex cover of size at most k. That is to say, is there a subset of vertices $V' \subseteq V$ such that for every edge $uv \in E$, $u \in V'$ or $v \in V'$.

Lemma 2. Vertex Cover *can be solved in time* $2^k \cdot n^{O(1)}$.

Proof. Consider an instance (G, k). Firstly, observe that any isolated vertices of G may be removed, since an isolated vertex cannot cover any edges. We now describe a branching algorithm. Consider an edge xy. Either x is in the vertex cover, or y is, giving a two-way branching. When branching on x, assume x is in the vertex cover, and remove x and all edges containing x from G. Continue branching at most k times, one of the leaves will either give a vertex cover, or the instance is a No-instance. The tree has at most 2^k leaves. Thus, the algorithm has running time $2^k \cdot n^{O(1)}$. □

To obtain a kernel, the algorithm applies a reduction based on when a vertex must necessarily be in the vertex cover.

Lemma 3. Vertex Cover *has a kernel with at most* k^2 *edges.*

Proof. Consider an instance (G, k). Firstly, observe that any isolated vertices of G may be removed, since an isolated vertex cannot cover any edges. Also observe that if there is a vertex x of degree greater than k it must be in the vertex cover. For, suppose not. Then all of $N(x)$ would be in the vertex cover, but then the vertex cover would be too large. Hence, if there is a vertex cover of size k it must contain x. Remove x (and the edges containing x), and decrease k by one.

In the new instance formed (G', k'), each vertex has degree bounded by k'. Thus, each vertex can only cover k' edges. If the number of edges is greater than k'^2, the instance must be a No-instance. Otherwise, the number of edges is bounded by $k'^2 \leq k^2$, as claimed. □

1.1 Parameterizations above/below a bound

In this section, we give an introduction to parameterizations above and below tight bounds (ATLB/BTLB). Consider the problem Max Acyclic Subgraph. In this problem, one is given a digraph $D = (V, A)$, and one aims to find an acyclic subdigraph of D with the maximum number

of arcs. Lets firstly consider the parameterized problem formed by asking whether D contains an acyclic subdigraph with at least k arcs. Assign an arbitrary ordering to the vertices of the digraph, and consider two arc sets, A', the set of arc going forward in this ordering, and A'', the set of backwards arcs. Observe $A = A' \cup A''$, each of these sets induce acyclic subdigraphs, and so D certainly has an acyclic subdigraph of size at least $|A|/2$. Hence if, $k \leq |A|/2$, the instance is a YES-instance, and if $k > |A|/2$ then $|V| \leq |A| + 1 \leq 2k$, and so the problem has a linear kernel. This problem has an $|A|^{O(1)}(2k)!$-time algorithm. To see this, for each of the $|V|! \leq (2k)!$ orderings of V, and construct the subdigraphs of D induced by the forward arcs, and check whether this forms an acyclic subgraph of the required size.

The lower bound motivates the study of a different parameterization of this problem:

MAX ACYCLIC SUBGRAPH-ATLB
Instance: A digraph $D = (V, A)$ and a nonnegative integer k.
Parameter: The integer k.
Question: Does there exist an acyclic subdigraphs with at least $\|A\|/2 + k$ arcs?

Gutin *et al.* [50] have proved that MAX ACYCLIC SUBGRAPH-ATLB is fixed-parameter tractable.

More generally, if one is given a maximization problem, that is to find a solution maximizing the value of some function P, and it is known P always achieves some value $P_{\min}$ for any assignment, then one can parameterize above this above the bound by asking if there is an assignment giving $P \geq P_{\min} + k$. However, the parameterization asking for an assignment giving $P \geq k$ may still be of interest, depending on the problem under consideration.

Problems parameterized above tight lower bounds were first considered by Mahajan and Raman [85]. They observed that often such parameterizations are of practical value. More recently, such problems have gained more interest, with Mahajan *et al.* [87] proving several results for problems, and also stating several open problems. Parameterizations above lower bounds are often resistant to traditional techniques in parameterized complexity. Gutin et. al. [49] developed a new probabilistic methodology for such problems.

1.2 MaxLin2

An important problem we will study is that of MAXLIN2. The problem is of interest for two reasons - it is interesting as a problem in it's own right, and it is a useful tool in our study of many other constraint satisfaction problems. Usefully, it can be related to the parameterization of other constraint satisfactions above average, where the lower bound is non-trivial and unwelcoming to direct attacks. In particular, it can be used as a tool for the widely studied problem of MAXSAT.

In the problem MAXLIN2 we are given a system of m linear equations in n variables over $\mathbb{F}_2$,

$x_1, \ldots, x_n$. The task is to maximize the total weight of satisfied equations. The problem MAX-r-LIN2 has the additional constraint that each equation contains at most r variables (we may allow r to be a function of n, if this is the case, $r(n)$ is used for clarity). This form of the problem is useful when reducing from MAX-r-SAT. Håstad [61] highlighted the importance of MAXLIN2 and MAX-r-LIN2 by stating that they are "as basic as satisfiability". For the application of MAXLIN2 to other problems, see, e.g., [3, 4, 23, 25, 61, 70].

The problem MAXLIN2 has previously received attention in the study of approximation algorithms (see [61, 62]). In this thesis, we consider the problem from a parameterized perspective. We firstly state the general form for the problem:

MAXLIN2

Instance: A system S of m linear equations in n variables over $\mathbb{F}_2$, where equation j is assigned a positive integral weight w_j, $j = 1, \ldots, m$, and a nonnegative integer k. We will write equation j in S as $\sum_{i \in \alpha_j} z_i = b_j$, where $\emptyset \neq \alpha_j \subseteq \{1, 2, \ldots, n\}$ and $|\alpha_j| = r_j$.

Parameter: k.

Task: Find an assignment of values to the n variables maximizing the weight of satisfied equations.

This formulation is the natural formulation for the problem when considered in isolation. However, it becomes convenient to consider the problem phrased with the equations in product form when making use of MAXLIN2 to solve other problems:

MAXLIN2

Instance: A system S of equations $\prod_{i \in \alpha_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $j = 1, \ldots, m$ and where each equation is assigned a positive integral weight w_j and $|\alpha_j| = r_j$.

It is clear from inspection that the two formulations are equivalent ($x_i = 1$ if and only if $z_i = 0$).

We now draw our attention to the matter of the choice of parameterization for this problem. Define $W = \sum_{j=1}^{m} w_j$. Then it is certainly true that equations of total weight at least $W/2$ equations can be satisfied simultaneously - the expected weight of equations satisfied by a uniform random assignment is $W/2$, and an algorithm using the method of conditional expectations would achieve such an assignment. We now observe that furthermore, this bound is tight. Consider a system consisting of pairs of equations of the form $\prod_{i \in I} x_i = -1$, $\prod_{i \in I} x_i = 1$, each equation in the pair being of the same weight. Any assignment to the variables will satisfy exactly one equation from each pair, so at most $W/2$ equations can be satisfied simultaneously.

We will let $\mathrm{sat}(S)$ denote the maximum total weight of equations that can be satisfied simultaneously.

MaxLin2-AA

Instance: A system S of equations $\prod_{i \in I_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $j = 1, \ldots, m$ and where each equation is assigned a positive integral weight w_j; and a nonnegative integer k.

Parameter: k.

Question: $\text{sat}(S) \geq W/2 + k$?

When the number of variables in each equation is bounded by r, the problem is stated as follows:

Max-r-Lin2-AA

Instance: A system S of equations $\prod_{i \in I_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $|I_j| \leq r$, $j = 1, \ldots, m$; equation j is assigned a positive integral weight w_j, and a nonnegative integer k.

Parameter: k.

Question: $\text{sat}(S) \geq W/2 + k$?

The best previously known results for MaxLin2 are those of Håstad. In particular, he showed an inapproximability result: for each $\epsilon > 0$ there is no polynomial time algorithm to distinguishing instances of Max-3-Lin in which at least $(1 - \epsilon)m$ equations can be simultaneously satisfied from instances in which less than $(1/2 + \epsilon)m$ equations can be simultaneously satisfied, unless P = NP. Hence, Max-r-Lin2-AA cannot be approximated within a constant factor $c > 1$ unless P = NP. Later work by Håstad and Venkatesh [62] showed that the problem does, however, admit a randomized polynomial-time algorithm which for any constant $c > 1$ outputs an assignment with approximation ratio at most $c^r\sqrt{m}$ with probability at least 3/4.

Fellows [38] and Niedermeier [92] introduced the notion of multivariate parameterized algorithms. Here, instead of having one parameter, k, the problem can be considered as having multiple parameters $k_1, k_2, \ldots, k_l$. We note that for the problem Max-r-Lin2-AA it would be interesting to consider the behaviour of the problem in both k and r. Thus we will consider both the problems MaxLin2-AA$[k]$, which only has k as the parameter, and Max-r-Lin2-AA$[k, r]$, where the problem is parameterized by both k and r. We observe that it is possible to reduce a problem with multiple parameters, $k_1, k_2, \ldots, k_l$ into a problem with one parameter by defining $k = k_1 + k_2 + \ldots + k_l$.

In Chapter 4 we show a kernelization algorithm for MaxLin2-AA$[k]$, admitting a kernel with at most $O(k^2 \log k)$ variables. Using a new notion of a sum-free subset of vectors we obtain an FPT algorithm for MaxLin2-AA$[k]$ with running time $2^{O(k \log k)}(nm)^{O(1)}$. For the problem Max-$r$-Lin2-AA$[k, r]$, Gutin *et al.* [49] gave a $n \leq m = O(9^r k^2)$ kernel. We show that, in fact, the

problem has a kernel with $n \leq (2k-1)r$, and as a consequence, that the maximization problem MAX-r-LIN2-AA is in APX if restricted to $m = O(n)$ for all fixed r, if the weight of each equation is bounded by a constant. This is in sharp contrast to the fact previously observed fact that MAX-r-LIN2-AA is not in APX for each $r \geq 3$.

In Chapter 5 we consider a different parameterization of MAXLIN2:

MAXLIN2-BW
Question: Is there an assignment of values to the n variables such that the total weight of the satisfied equations is at least $W - k$, where $W = w_1 + \cdots + w_m$?

This parameterization is of interest when one wishes to satisfy almost all but k equations of the system.

We show that if one imposes the additional constrain that each equation contains at most two variables, then this problem is FPT. If instead, one assumes each variable appears in at most two equations, the problem has a polynomial-time algorithm. However, even if each equation has exactly three variables, and each variables appears in exactly three equations, the problem is $W[1]$-hard.

1.3 Pseudo-Boolean Functions

Optimization of pseudo-boolean function, that is to say a function $f : \{-1,1\}^n \to \mathbb{R}$ (see Section 2.2 for a full definition), is a useful, applicable area of research. Such functions arise in statistical mechanics, reliability theory, and manufacturing, for example, see the survey of Boros and Hammer [14]. In classic analysis, lower bounds on the maxima of trigonometric Fourier expansions, see [15]. We will focus our attention on applications in computer science and discrete maths (see [3, 94] other uses in computer science).

In Fourier analysis, the Boolean domain is often assumed to be $\{-1,1\}^n$ rather than more usual $\{0,1\}^n$ and we will follow this assumption here. Here we use the following well-known and easy to prove fact (see, e.g., [94]): each function $f : \{-1,1\}^n \to \mathbb{R}$ can be uniquely written as

$$f(x) = \hat{f}(\emptyset) + \sum_{I \in \mathcal{F}} \hat{f}(I) \prod_{i \in I} x_i. \tag{1.1}$$

where $\mathcal{F} \subseteq \{I : \emptyset \neq I \subseteq [n]\}$, $[n] = \{1, 2, \ldots, n\}$ and $\hat{f}(I)$ are non-zero reals. Formula (1.1) is the *Fourier expansion* of f and $\hat{f}(I)$ are the *Fourier coefficients* of f. The right hand side of (1.1) is a polynomial and the degree $\max\{|I| : I \in \mathcal{F}\}$ of this polynomial will be called the *degree* of f. Writing $\mathcal{F} = \{I_1, I_2, \ldots, I_{|\mathcal{F}|}\}$, let A be a $(0,1)$-matrix with n rows and $|\mathcal{F}|$ columns and with entries a_{ij} such that $a_{ij} = 1$ if and only if the term corresponding to I_j in (1.1) contains x_i.

In Section 4.4, we obtain the following lower bound on the maximum of a pseudo-boolean function f of degree r:

$$\max_x f(x) \geq \hat{f}(\emptyset) + \lfloor (\text{rank}A + r - 1)/r \rfloor \cdot \min\{|\hat{f}(I)| : I \in \mathcal{F}\}, \tag{1.2}$$

where rank$A$ is the rank of A over $\mathbb{F}_2$. (Note that since rank$A$ does not depend on the order of the columns in A, we may order the terms in (1.1) arbitrarily.)

To demonstrate the combinatorial usefulness of (1.2), we apply it to obtain a short proof of the well-known lower bound of Edwards-Erdős on the maximum size of a bipartite subgraph in a graph (the MAX CUT problem). Erdős [36] conjectured and Edwards [35] proved that every connected graph with n vertices and m edges has a bipartite subgraph with at least $m/2 + (n-1)/4$ edges. For short graph-theoretical proofs, see, e.g., Bollobás and Scott [12] and Erdős *et al.* [37]. We consider the BALANCED SUBGRAPH problem [8] that generalizes MAX CUT and show that our proof of the Edwards-Erdős bound can be easily extended to BALANCED SUBGRAPH. By contrast, the graph-theoretical proofs of the Edwards-Erdős bound do no seem to extend to the BALANCED SUBGRAPH problem.

Pseudo-boolean functions are a useful tool in the application of the MAXLIN2 to other problems. However, there are still advantages to be had by directly applying the methodology to a problem. We do this for the important problem of SATISFIABILITY, which we will consider in the next section.

1.4 Satisfiability

The problem of SATISFIABILITY is a fundamental problem in Computer Science. In it, one is given a CNF formula F with m clauses $(c_1, \ldots, c_m)$, and n variables $(x_1, \ldots, x_n)$. We will view a CNF formula as being a set of clauses, and thus $c_i \in F$ is a clause of the formula. The general aim is to find a truth assignment to the variables to satisfy the clauses. A *truth assignment* is a function $\tau : V(F) \to \{\text{TRUE, FALSE}\}$. A truth assignment τ *satisfies* a clause C if there exists $x \in V(F)$ such that $x \in C$ and $\tau(x) = \text{TRUE}$, or $\bar{x} \in C$ and $\tau(x) = \text{FALSE}$.

The class of problem where the aim is to maximise the number of satisfied clauses is known as MAXSAT. We state the general version of the problem, in a parameterized form, below:

MAXSAT$_f[k]$
Instance: A CNF formula F with clauses $c_1, \ldots, c_m$, and variables $x_1, \ldots, x_n$, and a nonnegative integer k. Clause c_i has r_i literals, $i = 1, \ldots, m$.
Parameter: k.
Task: Decide whether there is a truth assignment satisfying at least $f(F, k)$ clauses.

There are several natural questions that can be asked. The first is the task of finding an assignment satisfying at least k clauses. However, it is easy to see that for any CNF formula on m clauses, there exists an assignment to the variables satisfying at least $m/2$ clauses, by random

assignment. Using this observation Mahajan and Raman [85] showed that for this task, the problem is fixed-parameter tractable.

This lower bound generates a natural question. Let the task be to find an assignment satisfying $m/2+k$ clauses. Is this problem fixed-parameter tractable? Mahajan and Raman [85] showed that this problem is indeed fixed-parameter tractable. We note, however, that the bound is very rarely tight. A tight instance would consist of pairs of conflicting clauses (x) and $(\bar{x})$, for each variable x. It would be desirable to consider parameterizations of the problem where there are more tight instances, or alternatively instances where this case is removed.

In this thesis, we consider both types of such parameterizations. Observe that if a clause has r literals, then the probability it is satisfied by a uniform random assignment to the variables is $(1-1/2^r)$. Hence, if we assume clause c_i contains r_i literals, then by linearity of expectation, a uniform random assignment will satisfy at least $\text{asat}(F) = \sum_{i=1}^{m}(1-2^{-r_i})$ clauses. In the problem MaxSat-AA we ask for a truth assignment satisfying at least $\text{asat}(F)+k$ clauses:

MaxSat-AA
Task: Decide whether there is a truth assignment satisfying at least $\text{asat}(F)+k$ clauses, where $\text{asat}(F) = \sum_{i=1}^{m}(1-2^{-r_i})$.

In Chapter 6 we fully study the problem of MaxSat-AA. We show that Max-$r(n)$-Sat-AA is not fixed-parameter tractable unless P=NP for any $r(n) \geq \lceil \log n \rceil$. Also, we prove that unless the exponential time hypothesis (ETH) is false, Max-$r(n)$Sat-AA is not even in XP (that is to say that there is no algorithm with running time $(nm)^{O(f(k))}$, for some function f) for any $r(n) \geq \log\log n + \varphi(n)$, where $\varphi(n)$ is any real-valued unbounded strictly increasing computable function. These two results are proved in Section 6.2. These hardness results are complemented by positive results: we show that the lower bound above on $r(n)$ cannot be decreased much further as we prove that Max-$r(n)$Sat-AA is in XP for any $r(n) \leq \log\log n - \log\log\log n$ and fixed-parameter tractable for any $r(n) \leq \log\log n - \log\log\log n - \varphi(n)$, where $\varphi(n)$ is any real-valued unbounded strictly increasing computable function. This result generalizes the one of Alon *et al.* [3] and is proved in Section 6.3.

These results are in sharp contrast to the complexity status of the related problems MaxLin2-AA and Max-r-Sat-AA, which are known to be fixed-parameter tractable (Chapter 4). Also this is one of the very few problems in the 'above guarantee' parameterization world, which is known to be hard. See Mahajan *et al.* [87], for a number of other hard above guarantee problems.

The problem we study is one of the few problems in the 'above guarantee parameterization' where we parameterize above an instance-specific bound, as opposed to a generic bound, see Mahajan *et al.* [87] for a discussion on this issue. Another example of such parameterizations is the problem Vertex Cover parameterized above the maximum matching of the given graph. See [98, 30, 90, 83] for recent results on this problem.

In Chapter 7 we focus on a different lower bound, based on the matching number. Let B_F

denote the bipartite graph with partite sets $V(F)$ and F with an edge between $v \in V(F)$ and $c \in F$ if $v \in V(c)$. The *matching number* $\nu(F)$ of F is the size of a maximum matching in B_F. Clearly, $\mathrm{sat}(F) \geq \nu(F)$ and this lower bound for $\mathrm{sat}(F)$ is tight as there are formulas F for which $\mathrm{sat}(F) = \nu(F)$.

Hence the following parameterization is above a tight lower bound is interesting:

MaxSat-A$\nu(F)$
Instance: A CNF formula F and a positive integer α.
Parameter: $k = \alpha - \nu(F)$.
Question: Is $\mathrm{sat}(F) \geq \alpha$?

In Chapter 7, MaxSat-A$\nu(F)$ will be proved fixed-parameter tractable, but unlike other parameterizations considered so far, MaxSat-A$\nu(F)$ will be shown to have no polynomial-size kernel unless coNP$\subseteq$NP/poly, which is highly unlikely [9]. The main result shows that MaxSat-A$\nu(F)$ is fixed-parameter tractable by obtaining an algorithm with running time $O((2e)^{2k+O(\log^2 k)}(n+m)^{O(1)})$, where e is the base of the natural logarithm. We also develop a randomized algorithm for MaxSat-A$\nu(F)$ of expected runtime $O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})$.

The *deficiency* $\delta(F)$ of a formula F is $|F|-|V(F)|$; the *maximum deficiency* $\delta^*(F) = \max_{F' \subseteq F} \delta(F')$. A formula F is called *variable-matched* if $\nu(F) = |V(F)|$. Our main result implies fixed-parameter tractability of MaxSat parameterized by $\delta(F)$ for variable-matched formulas F.

There are two related results: Kullmann [77] obtained an $O(n^{O(\delta^*(F))})$-time algorithm for solving MaxSat for formulas F with n variables and Szeider [103] gave an $O(f(\delta^*(F))n^4)$-time algorithm for the problem, where f is a function depending on $\delta^*(F)$ only. Note that we cannot just drop the condition of being variable-matched from our result and expect a similar algorithm: it is not hard to see that the satisfiability problem remains NP-complete for formulas F with $\delta(F) = 0$.

A formula F is *minimal unsatisfiable* if it is unsatisfiable but $F \setminus c$ is satisfiable for every clause $c \in F$. Papadimitriou and Wolfe [95] showed that recognition of minimal unsatisfiable CNF formulas is complete for the complexity class[1] D^P. Kleine Büning [71] conjectured that for a fixed integer k, it can be decided in polynomial time whether a formula F with $\delta(F) \leq k$ is minimal unsatisfiable. Independently, Kullmann [77] and Fleischner and Szeider [40] (see also [39]) resolved this conjecture by showing that minimal unsatisfiable formulas with n variables and $n+k$ clauses can be recognized in $n^{O(k)}$ time. Later, Szeider [103] showed that the problem is fixed-parameter tractable by obtaining an algorithm of running time $O(2^k n^4)$. Note that Szeider's results follow from his results mentioned in the previous paragraph and the well-known fact that $\delta^*(F) = \delta(F)$ holds for every minimal unsatisfiable formula F. Since every minimal unsatisfiable formula is variable-matched [2], our main result also implies fixed-parameter tractability of recognizing

[1] D^P is the class of problems that can be considered as the difference of two NP-problems; clearly D^P contains all NP and all co-NP problems

minimal unsatisfiable formula with n variables and $n+k$ clauses, parameterized by k.

In Chapter 8 we instead look at the problem formed after removing the tight examples to the $m/2$ bound. Recall, this bound is tight when F consists of pairs of *conflicting unit clauses* (x) and $(\bar{x})$. Since each truth assignment satisfies exactly one clause in each pair of conflicting unit clauses, it is natural to reduce F to the *unit-conflict free (UCF)* form by deleting all pairs of conflicting clauses. Let $\text{sat}(F)$ be the maximum number of clauses that can be satisfied by a truth assignment. If F is UCF, then Lieberherr and Specker [79] proved that $\text{sat}(F) \geq \hat{\varphi}m$, where $\hat{\varphi} = (\sqrt{5}-1)/2$ (golden ratio inverse), and that for any $\epsilon > 0$ there are UCF CNF formulae F for which $\text{sat}(F) < m(\hat{\varphi}+\epsilon)$. Yannakakis [109] gave a short probabilistic proof that $\text{sat}(F) \geq \hat{\varphi}m$ by showing that if the probability of every variable appearing in a unit clause being assigned TRUE is $\hat{\varphi}$ (here we assume that for all such variables x the unit clauses are of the form (x)) and the probability of every other variable being assigned TRUE is $1/2$, then the expected number of satisfied clauses is $\hat{\varphi}m$.

Since $\hat{\varphi}m$ rather than $m/2$ is an asymptotically tight lower bound for UCF CNF formulae, Mahajan and Raman [85] also introduced the following parameterization of MaxSAT:

MaxSat-UCF-A($\hat{\varphi}m$)
Instance: A UCF CNF formula F with m clauses.
Parameter: A nonnegative integer k.
Question: Decide whether $\text{sat}(F) \geq \hat{\varphi}m + k$.

Mahajan and Raman conjectured that MaxSat-UCF-A($\hat{\varphi}m$) is fixed-parameter tractable. To solve the conjecture in the affirmative, we show the existence of an $O(k)$-variable kernel for MaxSat-UCF-A($\hat{\varphi}m$). This result follows from our improvement of the Lieberherr-Specker lower bound.

1.5 The Edwards-Erdős and Poljak-Turzík Bounds

The last part of this thesis will consider problems parameterized above the Poljak-Turzík bound.

Given a connected graph G on n vertices and m edges, the Edwards-Erdős bound [35] states that G has a bipartite subgraph with at least $m/2 + (n-1)/4$ edges. Poljak and Turzík [96] extended this bound to λ-extendible properties ($0 < \lambda < 1$). Informally, the key feature a λ-extendible property Π has is that given a graph $G \in \Pi$, and an extra edge uv, and any set E^* of edges with one endpoint in $\{u, v\}$ and the other in $V(G)$, there exists a graph G' containing all of G, the edge uv and at least $\lambda|E^*|$ edges from E^*.

The unweighted version of the Poljak-Turzík bound [96] states that for any λ-extendible property Π, given a connected graph $G = (V, E)$ there exists a spanning subgraph $G' = (V', E') \in \Pi$

such that $|E'| \leq \lambda \cdot |E| + \frac{1-\lambda}{2}(|V| - 1)$.

Note that the Edwards-Erdős bound is the Poljak-Turzík bound for $\lambda = 1/2$. In this thesis we focus on problems where $\lambda = 1/2$, not making direct use of λ-extendibility.

1.6 Acyclic Subgraph

In Chapter 9 we consider the problem of finding the maximum acyclic subgraph in a directed graph. This problem is well-studied in the literature in graph theory, algorithms and their applications alongside its dual, the feedback arc set problem, see, e.g., Chapter 15 in [7] and references therein. This is true, in particular, in the area of parameterized algorithmics [18, 49, 54, 99].

Each directed graph D with m arcs has an acyclic subgraph with at least $m/2$ arcs. To obtain such a subgraph, order the vertices $x_1, \ldots, x_n$ of D arbitrarily and consider two spanning subgraphs of D: D' with arcs of the form $x_i x_j$, and D'' with arcs of the form $x_j x_i$, where $i < j$. One of D' and D'' has at least $m/2$ arcs. Moreover, $m/2$ is the largest size of an acyclic subgraph in every symmetric digraph S (in a symmetric digraph the existence of an arc xy implies the existence of an arc yx). Thus, it makes sense to consider the parameterization above the tight bound $m/2$: decide whether a digraph D contains an acyclic subgraph with at least $m/2 + k$ arcs, where k is the parameter. Mahajan *et al.* [87] and Raman and Saurabh [99] asked what the complexity of this problem is. For the case of oriented graphs (i.e., directed graphs with no directed cycles of length 2), Raman and Saurabh [99] proved that the problem is fixed-parameter tractable. A generalization of this problem to integer-arc-weighted digraphs (where $m/2$ is replaced by the half of the total weight of D) was proved to be fixed-parameter tractable in [49].

For oriented graphs, $m/2$ is no longer a tight lower bound on the maximum size of an acyclic subgraph. Poljak and Turzík [96] proved the following tight bound on the maximum size of an acyclic subgraph of a connected oriented graph D: $\frac{m}{2} + \frac{n-1}{4}$. To see that the bound is indeed tight consider a directed path $x_1 x_2 \ldots x_{2t+1}$ and add to it arcs $x_3 x_1, x_5 x_3, \ldots, x_{2t+1} x_{2t-1}$. This oriented graph H_t consists of t directed 3-cycles and has $2t + 1$ vertices and $3t$ arcs. Thus, $\frac{m}{2} + \frac{n-1}{4} = 2t$ and $2t$ is the maximum size of an acyclic subgraph of H_t: we have to delete an arc from every directed 3-cycle as the cycles are arc-disjoint.

Raman and Saurabh [99] asked to determine the parameterized complexity of the following problem: decide whether a connected oriented graph D has an acyclic subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + k$ arcs, where k is the parameter. Observe that we may replace k by $\frac{k}{4}$ to ensure that the parameter k is always integral. Therefore, the complexity of the Raman-Saurabh problem above is equivalent to that of the following parameterized problem.

Acyclic Subgraph above Poljak-Turzík Bound (ASAPT)
Instance: An oriented connected graph G with n vertices and m arcs.
Parameter: The integer k.
Question: Does G contain an acyclic subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + \frac{k}{4}$ arcs?

We show this problem is fixed-parameter tractable, giving a kernel with $O(k^2)$ vertices and $O(k^2)$ arcs. We do this by combining structural graph-theoretical and algorithmic approaches, using both one-way, and two-way reduction rules.

1.7 Signed Max Cut

We consider undirected graphs with no parallel edges or loops and in which every edge is labelled by $+$ or $-$. We call such graphs *signed graphs*, and edges, labelled by $+$ and $-$, *positive* and *negative* edges, respectively. The labels $+$ and $-$ are the *signs* of the corresponding edges. Signed graphs are well-studied due to their various applications and interesting theoretical properties, see, e.g., [19, 31, 44, 55, 59, 64, 110].

Let $G = (V, E)$ be a signed graph and let $V = V_1 \cup V_2$ be a partition of the vertex set of G (i.e., $V_1 \cap V_2 = \emptyset$). We say that G is (V_1, V_2)-*balanced* if an edge with both endpoints in V_1, or both endpoints in V_2 is positive, and an edge with one endpoint in V_1 and one endpoint in V_2 is negative; G is *balanced* if it is (V_1, V_2)-balanced for some partition V_1, V_2 of V (V_1 or V_2 may be empty).

In some applications, we are interested in finding a maximum-size balanced subgraph of a signed graph [19, 31, 64, 110]. We will call this problem Signed Max Cut. This problem is a generalization of Max Cut and as such is NP-hard (Signed Max Cut is equivalent to Max Cut when all edges of G are negative). Hüffner *et al.* [64] parameterized Signed Max Cut below a tight upper bound: decide whether $G = (V, E)$ contains a balanced subgraph with at least $|E| - k$ edges, where k is the parameter. Hüffner *et al.* [64] showed that this parameterized problem is fixed-parameter tractable (FPT) using a simple reduction to the Edge Bipartization Problem: decide whether an unsigned graph can be made bipartite by deleting at most k edges (k is the parameter). Using this result and a number of heuristic reductions, Hüffner *et al.* [64] designed a nontrivial practical algorithm that allowed them to exactly solve several instances of Signed Max Cut that were previously solved only approximately by DasGupta *et al.* [31].

In Chapter 10, we consider a different parameterization of Signed Max Cut:

Signed Max Cut ATLB
Instance: A connected signed graph G with n vertices and m edges.
Parameter: The integer k.
Question: Does G contain a balanced subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + \frac{k}{4}$ edges?

Note, that we use $\frac{k}{4}$ instead of just k to ensure that k is integral and $\mathsf{pt}(G) = \frac{m}{2} + \frac{n-1}{4}$ is a tight lower bound on the number of edges in a balanced subgraph of G (this fact was first proved by Poljak and Turzík [96], for a different proof, see [26]). Whilst the parameterization of Hüffner *et al.* of Max Cut ATLB is of interest when the maximum number of edges in a balanced subgraph H of G is close to the number of edges of G, Signed Max Cut ATLB is of interest when the maximum number of edges in H is close to the minimum possible value in a signed graph on n vertices and m edges. Thus, the two parameterizations treat the opposite parts of the Signed Max Cut "spectrum."

It appears that it is much harder to prove that Signed Max Cut ATLB is FPT than to show that the parameterization of Hüffner *et al.* of Signed Max Cut is. Indeed, Signed Max Cut ATLB is a generalization of the same parameterization of Max Cut (denoted by Max Cut ATLB) and the parameterized complexity of the latter was an open problem for many years (and was stated as an open problem in several papers) until Crowston *et al.* [29] developed a new approach for dealing with such parameterized problems (recall that Max Cut is a special case of Signed Max Cut when all edges are negative). This approach was applied by Crowston *et al.* [20] to solve an open problem of Raman and Saurabh [99] on maximum-size acyclic subgraph of an oriented graph. Independently, this problem was also solved by Mnich *et al.* [89] who obtained the solution as a consequence of a meta-theorem which shows that several graph problems parameterized above a lower bound of Poljak and Turzík [96] are FPT under certain conditions.

While the meta-theorem is for both unlabeled and labeled graphs, all consequences of the meta-theorem in [89] are proved only for parameterized problems restricted to unlabelled graphs. A possible reason is that one of the conditions of the meta-theorem requires us to show that the problem under consideration is FPT on a special family of graphs, called almost forests of cliques. The meta-theorem is useful when it is relatively easy to find an FPT algorithm on almost forests of cliques. However, for Signed Max Cut ATLB it is not immediately clear what an FPT algorithm would be even on a clique.

Our attempts to check that Signed Max Cut ATLB is FPT on almost forests of cliques led us to reduction rules that are applicable not only to almost forests of cliques, but to arbitrary instances of Signed Max Cut ATLB. Thus, we found two alternatives to prove that Signed Max Cut ATLB is FPT: with and without the meta-theorem. Since the first alternative required stating the meta-theorem and all related notions and led us to a slightly slower algorithm than the second alternative, we decided to use the second alternative.

We reduce an arbitrary instance of Signed Max Cut ATLB to an instance which is an almost

forest of cliques, but with an important additional property which allows us to make use of a slight modification of a dynamic programming algorithm of Crowston *et al.* [29] for Max Cut ATLB on almost forests of cliques.

Apart from showing that Max Cut ATLB is FPT, Crowston *et al.* [29] proved that the problem admits a kernel with $O(k^5)$ vertices. They also found a kernel with $O(k^3)$ vertices for a variation of Max Cut ATLB, where the lower bound used is weaker than the Poljak-Turzík bound. They conjectured that a kernel with $O(k^3)$ vertices exists for Max Cut ATLB as well. In the main result of Chapter 10, we show that Signed Max Cut ATLB, which is a more general problem, also admits a polynomial-size kernel and, moreover, our kernel has $O(k^3)$ vertices. Despite considering a more general problem than in [29], we found a proof which is shorter and simpler than the one in [29]; in particular, we do not use the probabilistic method. An $O(k^3)$-vertex kernel for Signed Max Cut ATLB does not immediately imply an $O(k^3)$-vertex kernel for Max Cut ATLB, but the same argument as for Signed Max Cut ATLB shows that Max Cut ATLB admits an $O(k^3)$-vertex kernel. This confirms the conjecture above.

1.8 Summary of Results

We conclude this section with a summary of the main results contained in this thesis. In the constraints, r is the number of variables in each equation/clause, and s is the number of equations any variable appears in. Some of the results are subject to reasonable complexity-theoretic assumptions, and $\varphi(n)$ is any real-valued unbounded strictly increasing computable function of n.

Family	Problem	Constraint	Result
MaxLin2	-AA$[k]$		$O(k^2 \log k)$ vars kernel
	-AA$[k, r]$		$(2k-1)r$ vars kernel
	-BW	$r \leq 2$	FPT
	-BW	$s \leq 2$	Polynomial Time
	-BW	$r = s = 3$	$W[1]$-hard
MaxSAT	-AA	$r \leq \log\log n$ $- \log\log\log n - \varphi(n)$	FPT
	-AA	$r \leq \log\log n - \log\log\log n$	XP
	-AA	$r \geq \log\log n + \varphi(n)$	not in XP
	-AA	$r \geq \lceil \log n \rceil$	para-NP-complete
	-A$\nu(F)$		FPT, no poly kernel
	-UCF-A$(\hat{\varphi}m)$		FPT, $O(k)$ vars kernel
Above Poljak-Turzík	Acyclic Subgraph		$O(k^2)$ vertices and $O(k^2)$ arcs kernel
	Signed Max Cut		$O(k^3)$ vertex kernel

1.9 Bibliographic Notes

Most of our results have been previously published, either in conference proceedings, or in a journal, as listed below:

[23] Note on Max Lin-2 above average.
R. Crowston, G. Gutin, and M. Jones
Information Processing Letters 110: 451–454, 2010.

[25] Systems of linear equations over $\mathbb{F}_2$ and problems parameterized above average.
R. Crowston, G. Gutin, M. Jones, E. J. Kim, and I. Ruzsa.
Proc. SWAT 2010, Lect. Notes Comput. Sci. 6139 (2010), 164–175.

[26] Simultaneously Satisfying Linear Equations Over $\mathbb{F}_2$: MaxLin2 and Max-r-Lin2 Parameterized Above Average.
R. Crowston, M. Fellows, G. Gutin, M. Jones, F. Rosamond, S. Thomassé and A. Yeo.
Proc. FSTTCS 2011, LIPICS Vol. 13, 229–240.

[27] Parameterized Complexity of Satisfying Almost All Linear Equations over $\mathbb{F}_2$
R. Crowston, G. Gutin, M. Jones, A. Yeo
Theory of Computing Systems, June 2012

[21] Parameterized Complexity of MaxSat Above Average.
R. Crowston, G. Gutin, M. Jones, V. Raman and S. Saurabh
Proc. LATIN 2012, Lecture Notes in Computer Science 7256 (2012), 184–194.

[28] Fixed-Parameter Tractability of Satisfying Beyond the Number of Variables
R. Crowston, G. Gutin, M. Jones, V. Raman, S. Saurabh and A. Yeo
Algorithmica October 2012

[22] A new lower bound on the maximum number of satisfied clauses in Max-SAT and its algorithmic applications.
R. Crowston, G. Gutin, M. Jones, and A. Yeo
Algorithmica 64 (2012), 56–68.

[20] Directed Acyclic Subgraph Problem Parameterized above Poljak-Turzik Bound.
R. Crowston, G. Gutin and M. Jones
Proc. FSTTCS 2012, LIPICS Vol. 18, 400–411.

[24] Maximum balanced subgraph problem parameterized above lower bound.
R. Crowston, G. Gutin, M. Jones, and G. Muciaccia
Proc. COCOON 2013, to appear.

Chapter 2

Notation

In this section we define standard terminology and basic facts that will be useful throughout this thesis.

For an integer n, $[n]$ stands for $\{1, \ldots, n\}$. A function f is *strictly increasing* if for every pair x', x'' of values of the argument with $x' < x''$, we have $f(x') < f(x'')$.

2.1 Graph Theory

2.1.1 Graphs and Hypergraphs

An *(undirected) graph* G consists of a set of *vertices* $V(G)$, and a set of *edges* $E(G)$. An edge $uv \in E(G)$ is an unordered pair of vertices, and $u, v \in V(G)$ are *endpoints* of this edge. We say u is *adjacent* to v. Normally n will be used to denote the number of vertices $|V(G)|$, and m the number of edges. Often a graph is viewed as an ordered pair (V, E).

The *neigborhood*, $N_G(v)$ of $v \in V(G)$ is the set of all vertices adjacent to v. That is to say $N_G(v) = \{u \in V(G) : uv \in E(G)\}$. For a subset $X \subseteq V(G)$, $N_G(X)$ denotes the set of all neighbours of vertices in X, $N_G(X) = \{u \in V(G) : \exists v \in X, uv \in E(G)\} \setminus X$. When G is clear from the context, we write $N(X)$ instead of $N_G(X)$. A *path* in a graph G is a sequence of disjoint vertices, $v_0, v_1, \ldots, v_p$ and edges $v_0v_1, v_1v_2, \ldots, v_{p-1}v_p$. A graph is *connected* if there exists a path between any pair of vertices.

A *bipartite graph* G is a graph whose vertices can be split into two disjoint sets $V(G) = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, such that each edge has one endpoint in V_1 and one endpoint in V_2. A *matching* in a graph is a subset $M \subseteq E(G)$ such that no two edges share an endpoint. We say that a matching *saturates* all end-vertices of its edges. The *matching number* $\nu(G)$ of G is the size of a maximum matching.

We make use of the classical Hall's Marriage Theorem, which states a bipartite graph G has a matching saturating every vertex of V_1 if and only if $|N(X)| \geq |X|$ for every subset $X \subseteq V_1$. The

following lemma follows immediately from Hall's Marriage Theorem. To see this, add d vertices to V_2, each adjacent to every vertex in V_1.

Lemma 4. *Let $G = (V_1, V_2; E)$ be a bipartite graph, and suppose that for all subsets $X \subseteq V_1$, $|N(X)| \geq |X| - d$ for some $d \geq 0$. Then $\nu(G) \geq |V_1| - d$.*

We say that a bipartite graph $G = (V_1, V_2; E)$ is *q-expanding* if for all $V_1' \subseteq V_1$, $|N_G(V_1')| \geq |V_1'| + q$. Given a matching M, an *alternating path* is a path in which the edges belong alternately to M and not to M.

A *hypergraph* generalizes the concept of a graph to allow an edge to contain more than two vertices.

Formally, a hypergraph $H = (V(H), \mathcal{F})$ consists of a set of *vertices* $V(H)$ and a family $\mathcal{F} \subseteq \mathcal{P} \backslash \emptyset$ of nonempty subsets of $V(H)$ called *edges* of H. We often denote $\mathcal{F}$ as $E(H)$. Note that $\mathcal{F}$ may have *parallel* edges, i.e., copies of the same subset of $V(H)$. For any vertex $v \in V(H)$, and any $\mathcal{E} \subseteq \mathcal{F}$, $\mathcal{E}[v]$ is the set of edges in $\mathcal{E}$ containing v, $N[v]$ is the set of all vertices contained in edges of $\mathcal{F}[v]$, and the *degree* of v is $d(v) = |\mathcal{F}[v]|$. For a subset T of vertices, $\mathcal{F}[T] = \bigcup_{v \in T} \mathcal{F}[v]$. If for each $e \in \mathcal{F}$, $|e| = k$, the hypergraph is *k-uniform*. In particular, a 2-uniform hypergraph where $\mathcal{F}$ is a set is simply a graph.

2.1.2 Directed Graphs

A *directed graph* D consists of a set of vertices $V(G)$, and a set of *arcs* $A(G)$. An arc $uv \in A(G)$ is an ordered pair of vertices. For a vertex x in D, the *out-degree* $d^+(x)$ is the number of arcs of D leaving x and the *in-degree* $d^-(x)$ is the number of arcs of D entering x. For a subset S of vertices of D, let $d^+(S)$ denote the number of arcs of D leaving S and $d^-(S)$ the number of arcs of D entering S. For subsets A and B of vertices of D, let $E(A, B)$ denote the set of arcs with exactly one endpoint in each of A and B (in both directions). For a set S of vertices, $D[S]$ is the subgraph of D induced by S. When $S = \{s_1, \ldots, s_p\}$, we will write $D[s_1, \ldots, s_p]$ instead of $D[\{s_1, \ldots, s_p\}]$. The *underlying graph* $\mathrm{UN}(D)$ of D is the undirected graph obtained from D by replacing all arcs by edges with the same end-vertices and getting rid of one edge in each pair of parallel edges. The *connected components* of D are connected components of $\mathrm{UN}(D)$; D is *connected* if $\mathrm{UN}(D)$ is connected. Vertices x and y of D are *neighbors* if there is an arc between them.

2.1.3 Treewidth

A *tree decomposition* of an (undirected) graph G is a pair (U, T) where T is a tree whose vertices we will call *nodes* and $U = (\{U_i \mid i \in V(T)\})$ is a collection of subsets of $V(G)$ such that

1. $\bigcup_{i \in V(T)} U_i = V(G)$,
2. for each edge $vw \in E(G)$, there is an $i \in V(T)$ such that $v, w \in U_i$, and
3. for each $v \in V(G)$ the set $\{i : \ v \in U_i\}$ of nodes forms a subtree of T.

The U_i's are called *bags*. The *width* of a tree decomposition $(\{U_i : i \in V(T)\}, T)$ equals $\max_{i \in V(T)}\{|U_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. We use notation $\mathbf{tw}(G)$ to denote the treewidth of a graph G.

2.2 Pseudo-Boolean Functions

A *pseudo-boolean function* is a function from $\{-1, 1\}^n$ to the set of reals. In Fourier analysis, it is convenient to assume the boolean domain is $\{-1, 1\}^n$ rather than the more usual $\{0, 1\}^n$, and we will follow this assumption here.

It is well-known and easy to prove (see, e.g., [94]) that a pseudo-boolean function, $f : \{-1, 1\}^n \to \mathbb{R}$ can be uniquely written as:

$$f(x) = \hat{f}(\emptyset) + \sum_{I \in \mathcal{F}} \hat{f}(I) \prod_{i \in I} x_i. \tag{2.1}$$

where $\mathcal{F} \subseteq \{I : \emptyset \neq I \subseteq [n]\}$. The terms $\hat{f}(I) \in \mathbb{R}$ are known as the *Fourier coefficients* of f, and formula 2.1 is the *Fourier expansion* of f.

Fourier analysis of pseudo-boolean functions has been used in many areas of computer science, see, e.g. [3, 25, 94].

2.3 Fixed-Parameter Tractability

The definition of fixed-parameter tractability is motivated by the desire to classify NP-complete problems recognising that some NP-hard problems are solvable in time polynomial in the instance size but superpolynomial in some parameter k. If the parameter is small, as often it may well be in a practical instance of a problem, than the problem is practically tractable. Thus, it is desirable to study such parameters.

We now proceed to give the formal definition. There are two (equivalent) systems of notation used in defining fixed-parameter tractability in the literature. Each formation can feel natural, depending on the problems under consideration. We give both definitions for completeness.

We firstly give the definitions in the notation of Downey and Fellows [33].

A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$ where Σ is a finite alphabet. The second component is called the *parameter* of the problem. L is *fixed-parameter tractable* if it can be determined in time $f(k) \cdot n^{O(1)}$ whether or not $(x, k) \in L$, where f is a computable function only depending on k.

Note that the Downey-Fellows notation implicitly makes the parameter part of the problem. Often, the parameter may be a structural property of the instance. Flum and Grohe [41] capture this by making their parameter a function of the problem. They denote a *parameterization* as a mapping $\kappa : \Sigma^* \to \mathbb{N}$ that is polynomial-time computable. A *parameterized problem* is denoted by a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ of strings over Σ and a parameterization κ of Σ. A

problem (Q, κ) is *fixed-parameter tractable* if there is an algorithm that decides Q with running time $f(\kappa(x)) \cdot |x|^{O(1)}$, where f is a computable function.

One can observe that these definitions capture the same concept. Replacing Σ^* with $\Sigma^* \times \mathbb{N}$ in Flum-Grohe, and defining $\kappa(x, k) = k$, one obtains a problem in Downey-Fellows.

Due to the problems being considered, we will use the Downey-Fellows notation.

We can now observe that for each parameterized problem, there is a corresponding nonparameterized version, formed by taking k as part of the input. If the nonparameterized problem is NP-hard, then $f(k)$ must be superpolynomial, provided P $\neq$ NP. However, this function might show the problem is practically tractable for small values of k.

A problem may have several parameters $k_1, \ldots, k_t$. In which case, it can be reduced to the one parameter case by setting $k = k_1 + \cdots + k_t$, see, e.g., [32].

2.3.1 Kernelization

Given a parameterized problem L , an instance $(x, k) \in L$, a *kernelization* is an algorithm that maps (x, k) to (x', k') in time polynomial in $|x|$ and k such that $(x, k) \in L$ if and only if $(x', k') \in L$, $|x'| \leq f(k)$ and $k' \leq g(k)$, for some computable functions f, g. The *size* of the kernel is $f(k)$ and the new instance (x', k') is known as the *kernel*. The kernelization algorithm should run in time polynomial in $|x|$ and k.

2.3.2 Bikernelization

The notion of a bikernelization was introduced in [3], it is useful in situations where there may be a natural, related problem that the problem under consideration may be transformed into. We relax the requirement that the algorithm maps to instances of the same problem, L, instead allowing a mapping to a different problem, L'. Such results are sometimes called compression results.

Formally, given parameterized problems L and L' a *bikernelization from L to L'* is an algorithm that maps (x, k) to (x', k') in time polynomial in $|x|$ and k such that $(x, k) \in L$ if and only if $(x', k') \in L'$, $|x'| \leq f(k)$ and $k' \leq g(k)$, for some computable functions f, g. The *size* of the bikernel is $f(k)$ and the new instance (x', k') is known as the *bikernel*. The kernelization algorithm should run in time polynomial in $|x|$ and k. Alon *et al.* [3] observed that a parameterized problem L is fixed-parameter tractable if and only if it is decidable and admits a bikernelization to a parameterized problem L'.

2.3.3 Parameterized Complexity classes

Recall for a problem L to be in FPT, the decision problem must be solvable in $f(k)|x|^{O(1)}$ time. If this requirement is weakened to the question of deciding membership in time $|x|^{O(f(k))}$, then L belongs to the parameterized complexity class XP.

It is known that FPT is a proper subset of XP [33]. Analogues of NP are provided by the classes of parameterized problems of the $W[t]$ Hierarchy giving the tower:

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

Here W[P] is the class of all parameterized problems (x, k) that can be decided in $f(k)|x|^{O(1)}$ time by a nondeterministic Turing machine that makes at most $f(k) \log |x|$ nondeterministic steps for some computable function f. For the definition of the classes W[t], see, e.g., [41]. It is believed that W[1] $\neq$ FPT, and thus showing a problem is W[1]-hard shows that it is unlikely to be FPT.

We will also make use of the exponential time hypothesis of Impagliazzo and Paturi [65]. The exponential time hypothesis is stronger than the P $\neq$ NP assumption, stating that 3-SAT cannot be solved in subexponential time.

Finally, we introduce the notion of para-NP-completeness. If a problem is para-NP-complete, there is no FPT algorithm unless P = NP. Let L and L' be parameterized problems with parameters k and k', respectively. An *fpt-reduction R from L to L'* is a many-to-one transformation from L to L', such that (i) $(I, k) \in L$ if and only if $(I', k') \in L'$ with $k' \leq g(k)$ for a fixed function g, and (ii) R is of complexity $O(f(k)|I|^c)$. Here (I', k') is the image of instance (I, k).

L is in *para-NP* if membership of (I, k) in L can be decided by a nondeterministic Turing machine in time $O(f(k)|I|^c)$, where $|I|$ is the size of I, $f(k)$ is an arbitrary function of the parameter k only, and c is a constant independent from k and I. A parameterized problem L' is *para-NP-complete* if it is in para-NP and for any parameterized problem L in para-NP there is an fpt-reduction from L to L'. It is well-known that a parameterized problem L belonging to para-NP is para-NP-complete if we can reduce an NP-complete problem to the subproblem of L when the parameter is equal to some constant [41].

For example, consider the k-Colorability problem, where given a graph G and a positive integer k (k is the parameter), we are to decide whether G is k-colorable. Since the (unparameterized) Colorability problem is in NP, k-Colorability is in para-NP. k-Colorability is para-NP-complete since 3-Colorability is NP-complete.

2.4 CNF formulas

We now define the notion of a boolean formula in *conjunctive normal form* (CNF). Recall that given a set of variables $x_1, ..., x_n$ for each variable x_i we have a positive literal x_i and a negative literal $\bar{x}_i$. A disjunction, $A \vee B$ is true if either A or B is true. A conjunction $A \wedge B$ is true if both A and B are true. We say a boolean formula is in conjunctive normal form if it is expressed as a conjunction of clauses, and each clause is a disjunction of literals. We write such a formula F as,

$$F = \bigwedge_{i=1}^{m} c_i$$

where each c_i is a disjunction of literals. We refer to c_i as the *clauses* of the formula.

For a subset X of the variables of CNF formula F, F_X denotes the subset of F consisting of all clauses c such that $V(c) \cap X \neq \emptyset$. A formula F is called *q-expanding* if $|X| + q \leq |F_X|$ for each

$X \subseteq V(F)$. Note that, by Hall's matching theorem, a formula is variable-matched if and only if it is 0-expanding. Clearly, a formula F is q-expanding if and only if B_F is q-expanding.

For $x \in V(F)$, $n(x)$ and $n(\bar{x})$ denote the number of clauses containing x and the number of clauses containing $\bar{x}$, respectively.

A function $L : U \to \{\text{TRUE}, \text{FALSE}\}$, where U is a subset of $V(F)$, is called a *partial truth assignment*. A partial truth assignment $L : U \to \{\text{TRUE}, \text{FALSE}\}$ is an *autarky* if L satisfies all clauses of F_U. We have the following:

Lemma 5 ([22]). *Let $L : U \to \{\text{TRUE}, \text{FALSE}\}$ be an autarky for a CNF formula F and let γ be any truth assignment on $V(F) \setminus U$. Then for the combined assignment $\tau := L \cup \gamma$, it holds that $\text{sat}_\tau(F) = |F_U| + \text{sat}_\gamma(F \setminus F_U)$. Clearly, τ can be constructed in polynomial time given L and γ.*

Autarkies were first introduced in [88]; they are the subject of much study, see, e.g., [39, 78, 103], and see [72] for an overview.

Part I

Parameterized Complexity of MaxLin2

Chapter 3

Motivating Results

3.1 Introduction

In this chapter we consider some special cases of MaxLin2, showing fixed parameter tractability using only combinatorial tools. This problem had previously been considered by Gutin *et al.* [50] using a probabilistic method. In this chapter, we extend the special cases considered using only combinatorial arguments.

Most of the results in this chapter will be strengthened using more advanced techniques in later chapters. The intention here is to introduce both the framework for solving MaxLin2 as well as how to relate it to other constraint satisfaction problems. In this chapter we will use the 'sum' notation for MaxLin2.

3.2 Max $r(n)$-Lin-2 above Average

Consider the following problem for a fixed function $r(n)$.

Max $r(n)$-Lin-2 above Average (or Max-$r(n)$Lin2-AA for short)
Instance: A system S of m linear equations in n variables over $\mathbb{F}_2$, where no equation has more than $r = r(n)$ variables and equation j is assigned a positive integral weight w_j, $j = 1, \ldots, m$, and a nonnegative integer k. We will write equation j in S as $\sum_{i \in \alpha_j} z_i = b_j$, where $\emptyset \neq \alpha_j \subseteq \{1, 2, \ldots, n\}$ and $\lvert\alpha_j\rvert \leq r$.
Parameter: The integer k.
Question: Is there an assignment of values to the n variables such that the total weight of the satisfied equations is at least $(W + k)/2$, where $W = w_1 + \cdots + w_m$?

We assume that each of the n variables appears in at least one equation of S.

Note that $W/2$ is indeed a tight lower bound for the above problem, as the expected weight of satisfied equations in a random assignment is $W/2$, and no assignment of values to the variables satisfies equations of total weight more than $W/2$ if S consists of pairs of equations with identical left-hand sides and contradicting right-hand sides.

Consider two reduction rules for MAX-$r(n)$LIN2-AA introduced in [50].

Reduction Rule 3.1. *Let A be the matrix of the coefficients of the variables in S, let $t =$ rank$A$ and let columns $a^{i_1}, \ldots, a^{i_t}$ of A be linearly independent. Then delete all variables not in $\{z_{i_1}, \ldots, z_{i_t}\}$ from the equations of S.*

Reduction Rule 3.2. *If we have, for a subset α of $\{1, 2, \ldots, n\}$, an equation $\sum_{i\in\alpha} z_i = b'$ with weight w', and an equation $\sum_{i\in\alpha} z_i = b''$ with weight w'', then we replace this pair by one of these equations with weight $w' + w''$ if $b' = b''$ and, otherwise, by the equation whose weight is bigger, modifying its new weight to be the difference of the two old ones. If the resulting weight is 0, we delete the equation from the system.*

Lemma 6. *[50] Let T be obtained from S by Rule 3.1 or 3.2. Then T is a* YES*-instance if and only if S is a* YES*-instance. Moreover, T can be obtained from S in time polynomial in n and m.*

If we cannot change S using Rule 3.1 (Rule 3.2), S is *irreducible* by Rule 3.1 (Rule 3.2). If S is irreducible by Rule 3.1, we have $n \leq m$. If S is irreducible by Rule 3.2, the symmetric difference $\alpha_j \Delta \alpha_p \neq \emptyset$ for each pair $j \neq p$.

Consider the following algorithm for MAX-$r(n)$LIN2-AA, which is a modification of an algorithm used in [62]. We assume that, in the beginning, no equation or variable in S is marked.

ALGORITHM $\mathcal{A}$

While $S \neq \emptyset$ and less than k equations are marked, do the following:

1. For $1 \leq i \leq n$, calculate ρ_i, the number of equations in S containing z_i.
2. Choose z_l with minimum ρ_l among all variables still in S. Mark z_l.
3. Choose an arbitrary equation containing z_l, $\sum_{i\in\alpha} z_i = b$.
4. Mark this equation and delete it from S.
5. Replace every equation $\sum_{i\in\alpha'} z_i = b'$ in S containing z_l by $\sum_{i\in\alpha\Delta\alpha'} z_i = b''$, where $b'' = b + b'$.
6. Apply Rule 3.2. (As a result, several equations can be of weight 0 and, thus, are deleted from the system.)

Observe that $\mathcal{A}$ runs in polynomial time. We have the following simple yet important property of $\mathcal{A}$.

Lemma 7. *If the input system S is irreducible by Rule 3.2 and algorithm $\mathcal{A}$ has marked k equations in S, then S is a* YES*-instance.*

Proof. Assume that $\mathcal{A}$ has marked k equations in the *input* system S and let T be the system of equations remaining in S after $\mathcal{A}$ has stopped. Observe that for every assignment of values to the variables $z_1, \ldots, z_n$ that satisfies all marked equations, the operation of Step 5 of $\mathcal{A}$ replaces S by an *equivalent* system (i.e., both systems have the same difference in weight of satisfied and falsified equations). Thus, for every such assignment, S is equivalent to T together with the marked equations. We will show that there is an assignment that satisfies all marked equations and half of the equations of T (in terms of weight). This will be sufficient due to the following. Let W' be the total weight of the marked equations. Then the total weight of the satisfied equations is $W' + (W - W')/2 = (W + W')/2 \geq (W + k)/2$ since $W' \geq k$ by integrality of the weights.

We can find a required assignment as follows. We start by finding an assignment of values to the variables in T that satisfies half of equations of T (in terms of weight), using the following algorithm from [62]: Assign values to the variables sequentially, and after each assignment, perform the obvious algebraic simplifications. When about to assign a value to z_j, consider all equations of the form $z_j = b$, for constant b. Assign z_j a value satisfying at least half of these equations (in terms of weight).

It remains to assign any values to the variables not in T of the marked equations such that they are all satisfied. This is possible if we find an assignment that satisfies the last marked equation, then find an assignment satisfying the equation marked before the last, etc. Indeed, the equation marked before the last contains a (marked) variable z_l not appearing in the last equation, etc. □

Lemma 8. *If an instance of* MAX-$r(n)$LIN2-AA *is irreducible by Rule 3.2 and its number of variables $n \geq 2^k r(n)$, then it is a* YES*-instance.*

Proof. Let ρ_{l_t} be the ρ_l picked in step 2 of Iteration t of algorithm $\mathcal{A}$, and let R_t be the maximum number of variables in any equation in S at Iteration t. Observe that $R_1 = r$, and that $R_{t+1} \leq 2R_t$. Thus, $R_t \leq 2^{t-1}r$.

In iteration t of $\mathcal{A}$ at most $2\rho_{l_t} - 1$ equations are removed - the selected equation containing z_l, and for each of the $\rho_{l_t} - 1$ equations two equations - the equation itself, and the equation it was cancelled out with by the application of Rule 3.2. Note that by the minimality of ρ_{l_t}, every variable appears in at least ρ_{l_t} equations. A variable is removed from the system if it is only contained in a removed equation, hence at most $(2\rho_{l_t} - 1)R_t/\rho_{l_t} < 2R_t$ variables will be removed from the system.

Thus, the total number of variables completely deleted from the system after $k - 1$ iterations is less than $\sum_{t=1}^{k-1} 2R_t \leq \sum_{t=1}^{k-1} 2^t r < 2^k r$. So, if $2^k r \leq n$ then Iteration k is possible, and hence, by Lemma 7, we have a YES-instance. □

Theorem 1. MAX-$r(n)$LIN2-AA *is fixed-parameter tractable if the instance is irreducible by Rule 3.2 and $r(n) = o(n)$.*

Proof. Let $r = o(n)$. By Lemma 8, if $n \geq 2^k r$, then we have a YES-instance. Otherwise, $n < 2^k r$ and so $n \leq g(k)$ for some function $g(k)$ depending on k only. In the last case, in time $O(m^{O(1)}2^{g(k)})$ we can check whether our instance is a YES-instance. □

Gutin *et al.* [50] prove that MAX-$r(n)$LIN2-AA is fixed-parameter tractable for $r = O(1)$. Using the method of [50] one can only extend this result to $r = o(\log m)$. If $r = o(\log m)$ then $r = o(n)$ (since $m < 2^n$ by Rule 3.2) and, thus, MAX-$r(n)$LIN2-AA is fixed-parameter tractable by Theorem 1. However, if $r = \Omega(\log m)$ and $r = o(n)$ then MAX-$r(n)$LIN2-AA is fixed-parameter tractable by Theorem 1, but this result cannot be obtained using the method of [50].

Recall that ρ_i is the number of equations in S containing z_i. Let $\rho = \max_{1 \leq i \leq n} \rho_i$. Let MAXLIN2-AA be MAX-$r(n)$LIN2-AA with $r(n) = n$.

Theorem 2. MAXLIN2-AA *is fixed-parameter tractable when the input system S with m equations is irreducible by Rules 3.1 and 3.2 and $\rho = o(m)$.*

Proof. Let $\rho = o(m)$. Apply algorithm $\mathcal{A}$ (for this theorem, there is no need to do Step 1 or select the z_l with minimum ρ_l on Step 2; we can arbitrarily choose any z_l still in S). We will show that after $k-1$ iterations at most $2\rho(k-1)$ equations have been deleted. Let Q be the set of equations that, at the beginning, contain at least one of $z_{l_1}, \ldots, z_{l_{k-1}}$, where $z_{l_1}, \ldots, z_{l_{k-1}}$ are the variables marked in the first $k-1$ iterations. Note that $|Q| \leq \rho(k-1)$. An equation not in Q is only deleted if there exists an equation in Q such that, after some applications of the symmetric difference operation of Step 5, the two equations have the same left-hand side. Furthermore, observe that each equation in Q can only ever have the same left-hand side as at most one equation not in Q. So the number of equations removed is at most $2|Q| \leq 2\rho(k-1)$. Observe that either $2\rho(k-1) < m$ in which case Iteration k is possible and we can apply Lemma 7, or $m \leq 2\rho(k-1)$ and therefore $m \leq f(k)$ for some function $f(k)$ depending on k only. If $m \leq f(k)$, $n \leq m \leq f(k)$ and in time $O(m^{O(1)}2^{f(k)})$ we can check whether our instance is a YES-instance. □

Using the approach of [50] it is easy to show that if $\rho = o(\sqrt{m})$ then MAXLIN2-AA is fixed-parameter tractable and this cannot be extended even to the case $\rho = \Theta(\sqrt{m})$. Thus, Theorem 2 provides a much stronger result.

3.3 Boolean Constraint Satisfaction Problems above Average

In this section we present a reduction from MAX-rSAT-AA (defined below) to MAXLIN2-AA, and prove that MAX-rSAT-AA is fixed-parameter tractable for constant r. The aim is to show that in the proofs of the main results of [3] for a wide family of Boolean Constraint Satisfaction Problems above Average, Lemma 8 can replace probabilistic and Fourier analysis inequalities. As a result, the proofs become purely combinatorial and slightly simpler. Alon *et al.* [3] provide all details for MAX-rSAT-AA only and comment that basically the same arguments can be used for a wide class of Boolean Constraint Satisfaction Problems above Average and, thus, we restrict ourselves to MAX-rSAT-AA only.

Let $r(\geq 2)$ be a constant.

MAX r-SAT ABOVE AVERAGE (or MAX-rSAT-AA for short)
Instance: A pair (F, k) where F is a multiset of m clauses, each of size r; F contains only variables $x_1, x_2, \ldots, x_n$, and k is a nonnegative integer.
Parameter: The integer k.
Question: Is there a truth assignment to the n variables such that the total number of satisfied clauses is at least $E + k2^{-r}$, where $E = m(1 - 2^{-r})$, the average number of satisfied clauses?

Let F contain clauses $C_1, \ldots, C_m$ in the variables $x_1, x_2, \ldots, x_n$. We may assume that $x_i \in \{-1, 1\}$, where -1 corresponds to TRUE. For F, consider a polynomial

$$X = \sum_{j=1}^{m} (1 - \prod_{x_i \in vars(C_j)} (1 + \epsilon_{(i,j)} x_i)),$$

where $vars(C_j)$ denotes the set of variables of C_j, that is, the variable $x_i \in C_j$ if and only if x_i `or` $\bar{x}_i \in C_j$, $\epsilon_i \in \{-1, 1\}$ and $\epsilon_{(i,j)} = 1$ if and only if the literal x_i is in C_j.

Lemma 9. *[3] The answer to* MAX-rSAT-AA *is* YES *if and only if there exists an assignment for* $x_1, x_2, \ldots, x_n$ *for which* $X \geq k$.

Theorem 3. *The problem* MAX-rSAT-AA *is fixed-parameter tractable for each constant* $r \geq 2$.

Proof. Given a MAX-rSAT-AA instance, define the polynomial X of degree at most r as above. After algebraic simplification $X = X(x_1, x_2, \ldots, x_n)$ can be written as $X = \sum_{I \in \mathcal{S}} X_I$, where $X_I = c_I \prod_{i \in I} x_i$, each c_I is a nonzero integer and $\mathcal{S}$ is a family of nonempty subsets of $\{1, \ldots, n\}$ each with at most r elements. Thus, X is a polynomial of degree at most r.

Now define an instance MAX-rLIN2-AA with the variables $z_1, z_2, \ldots, z_n$ as follows. For each nonzero term $c_I \prod_{i \in I} x_i$ consider the linear equation $\sum_{i \in I} z_i = b$, where $b = 0$ if c_I is positive, and $b = 1$ if c_I is negative, and assign this equation the weight $w_I = |c_I|$. It is easy to check that this system of equations has an assignment z_i satisfying equations of total weight at least $[\sum_{I \in \mathcal{S}} w_I + k]/2$ if and only if there are $x_i \in \{-1, 1\}$ so that $X(x_1, x_2, \ldots, x_n) \geq k$. This is shown by the transformation $x_i = (-1)^{z_i}$. Let n' be the number of variables in the instance of MAX-rLIN2-AA; clearly $n' \leq n$. Observe that $|S| \leq n^r$.

By Lemma 8, if $n' > 2^k r$, then we have a YES-instance of MAX-rLIN2-AA and, thus, by Lemma 9, the answer to MAX-rSAT-AA is YES. If $n' \leq 2^k r$ then we can find the maximum of X by using all assignments in time $|S|^{O(1)} 2^{n'} = n^{O(r)} 2^{r2^k}$ and apply Lemma 9 to check whether the answer to MAX-rSAT-AA is YES.

Finally, observe that the instance of MAX-rLIN2-AA can be constructed in time $(m2^r)^{O(1)}$. □

Chapter 4

MaxLin2 Parameterized Above Average

In this chapter we focus on two problems. We firstly consider MaxLin2-AA:

MaxLin2-AA (or MaxLin2-AA[k])

Instance: A system S of equations $\prod_{i \in I_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$ and where each equation is assigned a positive integral weight w_j, $W = \sum_{j \in [m]} w_j$; and a nonnegative integer k.

Parameter: k.

Question: $\mathrm{sat}(S) \geq W/2 + k$?

In situations where the parameter is clear, or we are studying one parameter, the $[k]$ suffix may be omitted. The first main result of this chapter is Theorem 6. Here we show that MaxLin2-AA[k], that is MaxLin2 parameterized above average by just k, admits a kernel with at most $O(k^2 \log k)$ variables. The number of equations may still be exponential in k, however, the number of variables is often the important factor when running an exact algorithm. As a consequence of this, we show that MaxLin2-AA[k] is fixed parameter tractable. The algorithm we present has running time $2^{O(k \log k)}(nm)^{O(1)}$.

The proof of Theorem 6 is based on two results: (a) If S is an irreducible system (i.e., a system that cannot be reduced using Rule 4.1 or 4.2 defined in Section 4.1) of MaxLin2-AA[k] and $2k \leq m \leq 2^{n/(2k-1)} - 2$, then S is a Yes-instance; (b) there is an algorithm for MaxLin2-AA[k] of complexity $n^{2k}(nm)^{O(1)}$. To prove (a), we introduce a new notion of a sum-free subset of vectors over $\mathbb{F}_2$ and show the existence of such subsets using linear algebra. We also prove that MaxLin2-AA[k] can be solved in time $2^{O(k \log k)}(nm)^{O(1)}$ (Corollary 1).

We then proceed to consider the problem restricted to the case when the number of variables in each equation is bounded by r:

MAX-r-LIN2-AA
Instance: A system S of equations $\prod_{i\in I_j} x_i = b_j$, where $x_i, b_j \in \{-1,1\}$, $\lvert I_j\rvert \leq r$, $j = 1,\ldots,m$; equation j is assigned a positive integral weight w_j, and a nonnegative integer k.
Parameter: k.
Question: $\mathrm{sat}(S) \geq W/2 + k$?

In the other main result of this chapter, Theorem 7, we give a sharp lower bound on the maximum excess for MAX-r-LIN2-AA as follows. Let S be an irreducible system and suppose that each equation contains at most r variables. Let $n \geq (k-1)r + 1$ and let $w_{\min}$ be the minimum weight of an equation of S. Then, in time $m^{O(1)}$, we can find an assignment x^0 to variables of S such that $\varepsilon_S(x^0) \geq k \cdot w_{\min}$. Essentially Theorem 7 follows from the existence of sum-free sets of vectors satisfying some simple conditions.

In Section 4.1, we give some reduction rules for MAX-r-LIN2-AA, describe an algorithm $\mathcal{B}$, based on Algorithm $\mathcal{A}$ In Chapter 3, and give some properties of the maximum excess, irreducible systems and Algorithm $\mathcal{B}$. In Section 4.2, we prove Theorem 6 and Corollary 1. A key tool in our proof of Theorem 7 is a lemma on sum-free subsets in a set of vectors from $\mathbb{F}_2^n$. The lemma and Theorem 7 are proved in Section 4.3. We prove several corollaries of Theorem 7 in Section 4.4. The corollaries are relevant to parameterized and approximation algorithms, as well as lower bounds for the maxima of pseudo-boolean functions and their applications in graph theory.

4.1 Maximum Excess, Irreducible Systems and Algorithm $\mathcal{B}$

Define the *excess* for $x^0 = (x_1^0,\ldots,x_n^0) \in \{-1,1\}^n$ over S to be

$$\varepsilon_S(x^0) = \sum_{j=1}^{m} c_j \prod_{i\in I_j} x_i^0, \quad \text{where } c_j = w_j b_j.$$

Note that $\varepsilon_S(x^0)$ is the total weight of equations satisfied by x^0 minus the total weight of equations falsified by x^0. The maximum possible value of $\varepsilon_S(x^0)$ is the *maximum excess* of S. Håstad and Venkatesh [62] initiated the study of the excess of a system of equations. In this chapter, we study the maximum excess for both MAXLIN2-AA and MAX-r-LIN2-AA. Note that the excess is a *pseudo-boolean function* [14], i.e., a function that maps $\{-1,1\}^n$ to the set of reals.

Remark 1. *Observe that the answer to* MAXLIN2-AA *is* YES *if and only if the maximum excess is at least* $2k$.

Remark 2. *The excess $\varepsilon_S(x)$ is a pseudo-boolean function and its Fourier expression is $\varepsilon_S(x) = \sum_{j=1}^{m} c_j \prod_{i\in I_j} x_i$. Moreover, observe that every pseudo-boolean function $f(x) = \sum_{I\in\mathcal{F}} \hat{f}(I) \prod_{i\in I} x_i$ (where $\hat{f}(\emptyset) = 0$) is the excess over the system $\prod_{i\in I} x_i = b_I$, $I \in \mathcal{F}$, where $b_I = 1$ if $\hat{f}(I) > 0$ and $b_I = -1$ if $\hat{f}(I) < 0$, with weights $|\hat{f}(I)|$. Thus, studying the maximum excess over a* MaxLin2-AA*-system (with real weights) is equivalent to studying the maximum of a pseudo-boolean function.*

Consider two reduction rules for MaxLin2 studied in [49], and previously stated in a different form In Chapter 3. Rule 4.1 was studied before in [62].

Reduction Rule 4.1. *If we have, for a subset I of $[n]$, an equation $\prod_{i\in I} x_i = b'_I$ with weight w'_I, and an equation $\prod_{i\in I} x_i = b''_I$ with weight w''_I, then we replace this pair by one of these equations with weight $w'_I + w''_I$ if $b'_I = b''_I$ and, otherwise, by the equation whose weight is bigger, setting its new weight to $|w'_I - w''_I|$. If the resulting weight is 0, we delete the equation from the system.*

Hereafter, rank$A$ will denote the rank of A over $\mathbb{F}_2$.

Reduction Rule 4.2. *Let A be the matrix over $\mathbb{F}_2$ corresponding to the set of equations in S, such that $a_{ji} = 1$ if $i \in I_j$ and 0, otherwise. Let $t =$ rank$A$ and suppose columns $a^{i_1}, \ldots, a^{i_t}$ of A are linearly independent. Then delete all variables not in $\{x_{i_1}, \ldots, x_{i_t}\}$ from the equations of S.*

Lemma 10. *[49] Let S' be obtained from S by Rule 4.1 or 4.2. Then the maximum excess of S' is equal to the maximum excess of S. Moreover, S' can be obtained from S in time polynomial in n and m.*

To see the validity of Rule 4.2, consider an independent set I of columns of A of cardinality rank$A$ and a column $a^j \notin I$. Observe that $a^j = \sum_{i\in I'} a^i$, where $I' \subseteq I$. Consider an assignment $z = z^0$. If $z^0_j = 1$ then for each $i \in I' \cup \{j\}$ replace z^0_i by $z^0_i + 1$. The new assignment satisfies exactly the same equations as the initial assignment. Thus, we may assume that $z_j = 0$ and remove z_j from the system. For a different proof, see [50]. If we cannot change a weighted system S using Rules 4.2 and 4.1, we call it *irreducible*.

Lemma 11. *Let S' be a system obtained from S by first applying Rule 4.1 as long as possible and then applying Rule 4.2. Then S' is irreducible.*

Proof. Let S^* denote the system obtained from S by applying Rule 4.1 as long as possible. Without loss of generality, assume that $x_1 \notin \{x_{i_1}, \ldots, x_{i_t}\}$ (see the description of Rule 4.2) and thus Rule 4.2 removes x_1 from S^*. To prove the lemma it suffices to show that after x_1 removal no pair of equations has the same left hand side. Suppose that there is a pair of equations in S^* which has the same left hand side after x_1 removal; let $\prod_{i\in I'} x_i = b'$ and $\prod_{i\in I''} x_i = b''$ be such equations and let $I' = I'' \cup \{1\}$. Then the entries of the first column of A, a^1, corresponding to the pair of equations are 1 and 0, but in all the other columns of A the entries corresponding to the the pair of equations are either 1,1 or 0,0. Thus, a^1 is independent from all the other columns of A, a contradiction.

Note that since the rank is unchanged, only one application of Rule 4.2 is required. □

Let S be an irreducible system of MaxLin2-AA. Consider the following algorithm. We assume that, in the beginning, no equation or variable in S is marked.

> Algorithm $\mathcal{B}$
>
> While the system S is nonempty do the following:
>
> 1. Choose an equation $\prod_{i\in I} x_i = b$ and mark a variable x_l such that $l \in I$.
> 2. Mark this equation and delete it from the system.
> 3. Replace every equation $\prod_{i\in I'} x_i = b'$ in the system containing x_l by $\prod_{i\in I\Delta I'} x_i = bb'$, where $I\Delta I'$ is the symmetric difference of I and I' (the weight of the equation is unchanged).
> 4. Apply Reduction Rule 4.1 to the system.

Note that algorithm $\mathcal{B}$ replaces $Az = b$ with an *equivalent* system under the assumption that the marked equations are satisfied; that is, for every assignment of values to the variables $z_1, \ldots, z_n$ that satisfies the marked equations, both systems have the same excess.

The *maximum $\mathcal{B}$-excess* of S is the maximum possible total weight of equations marked by $\mathcal{B}$ for S taken over all possible choices in Step 1 of $\mathcal{B}$. The following lemma indicates the potential power of $\mathcal{B}$.

Lemma 12. *Let S be an irreducible system. Then the maximum excess of S equals its maximum $\mathcal{B}$-excess. Furthermore, for any set of equations marked by Algorithm $\mathcal{B}$, in polynomial time, we can find an assignment of excess at least the total weight of marked equations.*

Proof. We first prove that the maximum excess of S is not smaller than its maximum $\mathcal{B}$-excess. By construction, for any assignment that satisfies all the marked equations, half of the non-marked equations by weight are satisfied. Therefore it suffices to find an assignment to the variables such that all marked equations are satisfied. Assign arbitrary values to the unmarked variables. Then assign values to the marked variables in the order opposite to which they were marked such that the corresponding marked equations are satisfied.

The above argument proves also the last statement of the lemma.

Now we prove that the maximum $\mathcal{B}$-excess of S is not smaller than its maximum excess. Let $x^0 = (x_1^0, \ldots, x_n^0)$ be an assignment that achieves the maximum excess, t. Observe that if at each iteration of $\mathcal{B}$ we mark an equation that is satisfied by x^0, then $\mathcal{B}$ will mark equations of total weight t. □

Remark 3. It follows from Lemma 12 that the maximum excess of a (nonempty) irreducible system $Az = b$ with smallest weight $w_{\min}$ is at least $w_{\min}$. If all weights are integral, then the maximum excess of $Az = b$ is at least 1.

Clearly, the total weight of equations marked by $\mathcal{B}$ depends on the choice of equations to mark in Step 1. In the next section we will use the notion of Sum-free sets to obtain a set of equations such that we can mark each equation in the set in successive iterations of $\mathcal{B}$. This means we can run $\mathcal{B}$ a guaranteed number of times, which we can use to get a lower bound on the $\mathcal{B}$-excess.

4.2 MaxLin2-AA

The following two theorems provide a basis for proving Theorem 6, the main result of this section.

Theorem 4. *There exists an $n^{2k}(nm)^{O(1)}$-time algorithm for* MaxLin2-AA*[k] that returns an assignment of excess of at least $2k$ if one exists, and returns* NO *otherwise.*

Proof. Suppose we have an instance $\mathcal{L}$ of MaxLin2-AA[k] that is reduced by Rules 4.1 and 4.2, and that the maximum excess of $\mathcal{L}$ is at least $2k$. Let A be the matrix introduced in Rule 4.2. Pick n equations $e_1, \ldots, e_n$ such that their rows in A are linearly independent. An assignment of excess at least $2k$ must either satisfy at least one of these equations, or falsify them all. If they are all falsified, then the system of equations $\bar{e}_1, \ldots, \bar{e}_n$, where each $\bar{e}_i$ is e_i with the changed right hand side, has a unique solution, an assignment of values to $x_1, \ldots, x_n$. If this assignment does not give excess at least $2k$ for $\mathcal{L}$, then any assignment that leads to excess at least $2k$ must satisfy at least one of $e_1, \ldots, e_n$. Thus, by Lemma 12, algorithm $\mathcal{H}$ can mark one of these equations and achieve an excess of at least $2k$.

This gives us the following depth-bounded search tree. At each node N of the tree, reduce the system by Rules 4.1 and 4.2, and let n' be the number of variables in the reduced system. Then find n' equations $e_1, \ldots, e_{n'}$ corresponding to linearly independent vectors. Find an assignment of values to $x_1, \ldots, x_{n'}$ that falsifies all of $e_1, \ldots, e_{n'}$. Check whether this assignment achieves excess of at least $2k - w^*$, where w^* is total weight of equations marked by $\mathcal{B}$ in all predecessors of N. If it does, then return the assignment and stop the algorithm. Otherwise, split into n' branches. In the i'th branch, run an iteration of $\mathcal{B}$ marking equation e_i. Then repeat this algorithm for each new node. Whenever the total weight of marked equations is at least $2k$, return the suitable assignment. Clearly, the algorithm will terminate without an assignment if the maximum excess of $\mathcal{L}$ is less than $2k$.

All the operations at each node take time $(nm)^{O(1)}$, and there are less than n^{2k+1} nodes in the search tree. Therefore this algorithm takes time $n^{2k}(nm)^{O(1)}$. □

The following lemma is used to prove Theorem 5, but it might be also of independent interest. Let M be a set of m vectors in $\mathbb{F}_2^n$ and let A be a $m \times n$-matrix in which the vectors of M are rows. Using Gaussian elimination on A one can find a maximum size linearly independent subset of the rows of M in polynomial time [74]. Let K and M be sets of vectors in $\mathbb{F}_2^n$ such that $K \subseteq M$. We say K is *M-sum-free* if no sum of two or more distinct vectors in K is equal to a vector in M. Observe that K is M-sum-free if and only if K is linearly independent and no sum of vectors in K is equal to a vector in $M \backslash K$.

Lemma 13. *Let M be a set in $\mathbb{F}_2^n$ such that M contains a basis of $\mathbb{F}_2^n$, the zero vector is in M and $|M| < 2^n$. If k is a positive integer and $k+1 \leq |M| \leq 2^{n/k}$ then, in time $|M|^{O(1)}$, we can find an M-sum-free subset K of $k+1$ vectors such that no sum of two or more vectors of K is in M.*

Proof. We first consider the case when $k = 1$. Since $|M| < 2^n$ and the zero vector is in M, there is a non-zero vector $v \notin M$. Since M contains a basis for $\mathbb{F}_2^n$ (which can be found in polynomial time using Gaussian elimination, see above), v can be written as a sum of vectors in M and consider such a sum with the minimum number of summands: $v = u_1 + \cdots + u_\ell$, $\ell \geq 2$. Since $u_1 + u_2 \notin M$, we may set $K = \{u_1, u_2\}$. We can find such a set K in polynomial time by looking at every pair in $M \times M$.

We now assume that $k > 1$. Since $k+1 \leq |M| \leq 2^{n/k}$ we have $n \geq k+1$.

We will now proceed with a greedy algorithm to find K. We firstly given an informal outline of the argument. The algorithm tries to find a set K by constructing a sum-free subset L greedily one vector at a time. If a set of size $k+1$ is obtained, the algorithm terminates. Otherwise, the set L is used to 'compress' the problem into one on M', a set in $\mathbb{F}_2^{n'}$, $n' < n$ satisfying the conditions of the lemma. Since n decreases each time, the algorithm terminates, and we show, in fact, it terminates in finding the set required. For the reader familiar with vector space terminology, $\mathbb{F}_2^{n'}$ is $\mathbb{F}_2^n$ modulo span(L), the subspace of $\mathbb{F}_2^n$ spanned by L, and M' is the image of M in $\mathbb{F}_2^{n'}$, however the proof that follows will not assume the reader knows this.

We now state the argument more formally. Suppose we have a set $L = \{a_1, \ldots, a_l\}$ of vectors in M, $l \leq k$, such that no sum of two or more elements of L is in M. We can extend this set to a basis, so $a_1 = (1, 0, 0, \ldots, 0)$, $a_2 = (0, 1, 0, \ldots, 0)$ and so on. For every $a \in M \backslash L$ we check whether $M \backslash \{a_1, \ldots, a_l, a\}$ has an element that agrees with a in all co-ordinates $l+1, \ldots, n$. If no such element exists, then we add a to the set L, as no element in M can be expressed as a sum of a and a subset of L.

If our greedy algorithm finds a set L of size at least $k+1$, we are done and L is our set K. Otherwise, we have stopped at $l \leq k$. In this case, we do the next iteration as follows. Recall that L is part of a basis of M such that $a_1 = (1, 0, 0, \ldots, 0)$, $a_2 = (0, 1, 0, \ldots, 0), \ldots$. We create a new set M' in $\mathbb{F}_2^{n'}$, where $n' = n - l$. We do this by removing the first l co-ordinates from M, and then identifying together any vectors that agree in the remaining n' co-ordinates. We are in effect identifying together any vectors that only differ by a sum of some elements in L. It follows that every element of M' was created by identifying together at least two elements of M, since otherwise we would have had an element in $M \backslash L$ that should have been added to L by our greedy algorithm. Therefore it follows that $|M'| \leq |M|/2 \leq 2^{n/k-1}$. From this inequality and the fact that $n' \geq n - k$, we get that $|M'| \leq 2^{n'/k}$. It also follows by construction of M' that M' has a basis for $\mathbb{F}_2^{n'}$, and that the zero vector is in M'. (Thus, we have $|M'| \geq n' + 1$.) If $n' \geq k+1$ we complete this iteration by running the algorithm on the set M' as in the first iteration. Otherwise ($n' \leq k$), the algorithm stops.

Since each iteration of the algorithm decreases n', the algorithm terminates. Now we prove that at some iteration, the algorithm will actually find a set K of $k+1$ vectors.

Suppose not, then the algorithm reaches the point when $1 \leq n' \leq k$. Observe that $|M'| \geq n'+1$ and $|M'| \leq 2^{n'/k}$. This implies $n'+1 \leq 2^{n'/k} \leq 2^{k/k} \leq 2$, which implies $n' = k = 1$, a contradiction with the assumption that $k > 1$.

It is easy to check that the running time of the algorithm is polynomial in $|M|$. □

Remark 4. It is much easier to prove a non-constructive version of the above result. In fact we can give a non-constructive proof that $k+1 \leq |M| \leq 2^{n/k}$ can be replaced by $2k < |M| < 2^{n/k}((k-1)!)^{1/k}$. We will extend our proof above for the case $k = 1$. We may assume that $k \geq 2$. Observe that the number of vectors of $\mathbb{F}_2^n$ that can be expressed as the sum of at most k vectors of M is at most

$$\binom{|M|}{k} + \binom{|M|}{k-1} + \cdots + \binom{|M|}{1} + 1 \leq |M|^k/(k-1)! \text{ for } |M| > 2k.$$

Since $|M| < 2^{n/k}((k-1)!)^{1/k}$ we have $|\mathbb{F}_2^n| > |M|^k/(k-1)!$ and, thus, at least for one vector a of $\mathbb{F}_2^n$ we have $a = m_1 + \cdots + m_\ell$, where ℓ is minimum and $\ell > k$. Note that, by the minimality of ℓ, no sum of two or more summands of the sum for a is in M and all summands are distinct. Thus, we can set $K = \{m_1, \ldots, m_{k+1}\}$.

Theorem 5. *Let S be an irreducible system of* MaxLin2-AA*[k] and let $k \geq 1$. If $2k \leq m \leq \min\{2^{n/(2k-1)} - 1, 2^n - 2\}$, then the maximum excess of S is at least $2k$. Moreover, we can find an assignment with excess of at least $2k$ in time $m^{O(1)}$.*

Proof. Consider a set M of vectors in $\mathbb{F}_2^n$ corresponding to equations in S as follows: for each equation $\prod_{i \in I} x_i = b$ in S, define a vector $v = (v_1, \ldots, v_n) \in M$, where $v_i = 1$ if $i \in I$ and $v_i = 0$, otherwise. Add the zero vector to M.

As S is reduced by Rule 4.2 and $k \leq m \leq \min\{2^{n/(2k-1)} - 1, 2^n - 2\}$, we have that M contains a basis for $\mathbb{F}_2^n$ and $k \leq |M| \leq \min\{2^{n/(2k-1)}, 2^n - 1\}$. Therefore, using Lemma 13 we can find an M-sum-free set K of $2k$ vectors. Let $\{e_{j_1}, \ldots, e_{j_{2k}}\}$ be the corresponding set of equations. Run Algorithm $\mathcal{B}$, choosing at Step 1 an equation of S from $\{e_{j_1}, \ldots, e_{j_{2k}}\}$ each time, and let S' be the resulting system. Algorithm $\mathcal{B}$ will run for $2k$ iterations of the while loop as no equation from $\{e_{j_1}, \ldots, e_{j_{2k}}\}$ will be deleted before it has been marked.

Indeed, suppose that this is not true. Then for some e_{j_l} and some other equation e in S, after applying Algorithm $\mathcal{B}$ for at most $l-1$ iterations e_{j_l} and e contain the same variables. Thus, there are vectors $v_j \in K$ and $v \in M$ and a pair of nonintersecting subsets K' and K'' of $K \setminus \{v, v_j\}$ such that $v_j + \sum_{u \in K'} u = v + \sum_{u \in K''} u$. Thus, $v = v_j + \sum_{u \in K' \cup K''} u$, contradicting the definition of K.

Thus, by Lemma 12, we are done. □

Theorem 6. *The problem* MaxLin2-AA*[k] has a kernel with at most $O(k^2 \log k)$ variables.*

Proof. Let $\mathcal{L}$ be an instance of MaxLin2-AA[k] and let S be the system of $\mathcal{L}$ with m equations and n variables. We may assume that S is irreducible. Let the parameter k be an arbitrary positive integer.

If $m < 2k$ then $n < 2k = O(k^2 \log k)$. If $2k \leq m \leq 2^{n/(2k-1)} - 2$ then, by Theorem 5 and Remark 1, the answer to $\mathcal{L}$ is YES and the corresponding assignment can be found in polynomial time. If $m \geq n^{2k} - 1$ then, by Theorem 4, we can solve $\mathcal{L}$ in polynomial time.

Finally we consider the case $2^{n/(2k-1)} - 2 \leq m \leq n^{2k} - 2$. Hence, $n^{2k} \geq 2^{n/(2k-1)}$. Therefore, $4k^2 \geq 2 + n/\log n \geq \sqrt{n}$ and $n \leq (2k)^4$. Hence, $n \leq 4k^2 \log n \leq 4k^2 \log(16k^4) = O(k^2 \log k)$.

Since S is irreducible, $m < 2^n$ and thus we have obtained the desired kernel. □

Corollary 1. *The problem* MaxLin2-AA*[k] can be solved in time* $2^{O(k \log k)}(nm)^{O(1)}$.

Proof. Let $\mathcal{L}$ be an instance of MaxLin2-AA[k]. By Theorem 6, in time $(nm)^{O(1)}$ either we solve $\mathcal{L}$ or we obtain a kernel with at most $O(k^2 \log k)$ variables. In the second case, we can solve the reduced system (kernel) by the algorithm of Theorem 4 in time $[O(k^2 \log k)]^{2k}[O(k^2 \log k)m]^{O(1)} = 2^{O(k \log k)}m^{O(1)}$. Thus, the total time is $2^{O(k \log k)}(nm)^{O(1)}$. □

Corollary 2. *Let $p(n)$ be a fixed function such that $p(n) = o(n)$. If $m \leq 2^{p(n)}$ then* MaxLin2-AA *is fixed-parameter tractable. Moreover, a satisfying assignment can be found in time $g(k)m^{O(1)}$ for some computable function g.*

Proof. We may assume that $m \geq n > k > 1$. Observe that $m \leq 2^{n/2k}$ implies $m \leq 2^{n/(2k-1)} - 2$. Thus, by Theorem 5, if $p(n) \leq n/2k$, the answer to MaxLin2-AA is YES, and there is a polynomial algorithm to find a suitable assignment. Otherwise, $n \leq f(k)$ for some function dependent on k only and MaxLin2-AA can be solved in time $2^{f(k)}m^{O(1)}$ by checking every possible assignment. □

Let ρ_i be the number of equations in $Az = b$ containing z_i, $i = 1, \ldots, n$. Let $\rho = \max_{i \in [n]} \rho_i$ and let r be the maximum number of variables in an equation of $Az = b$. Crowston *et al.* [23] proved that MaxLin2-AA is fixed-parameter tractable if either $r \leq r(n)$ for some fixed function $r(n) = o(n)$ or $\rho \leq \rho(m)$ for some fixed function $\rho(m) = o(m)$.

For a given $r = r(n)$, we have $m \leq \sum_{i=1}^{r} \binom{n}{i}$. By Corollary 23.6 in [67], $m \leq 2^{nH(r/n)}$, where $H(y) = -y \log_2 y - (1-y)\log_2(1-y)$, the entropy of y. It is easy to see that if $y = o(n)/n$, then $H(y) = o(n)/n$. Hence, if $r(n) = o(n)$, then $m \leq 2^{o(n)}$. By Corollary 23.5 in [67] (this result was first proved by Kleitman *et al.* [73]), for a given $\rho = \rho(m)$ we have $m \leq 2^{nH(\rho/m)}$. Therefore, if $\rho(m) = o(m)$ then $m \leq 2^{n \cdot o(m)/m}$ and, thus, $m \leq 2^{o(n)}$ (as $n \leq m$, if $n \to \infty$ then $m \to \infty$ and $o(m)/m \to 0$). Thus, both results of Crowston *et al.* [23] follow from corollary 2.

4.3 Max-rLin2-AA

We can easily prove Theorem 7 in the same way as we proved Theorem 5, but instead of Lemma 13, we use Lemma 14.

Lemma 14. *Let M be a set of vectors in $\mathbb{F}_2^n$ such that M contains a basis of $\mathbb{F}_2^n$. Suppose that each vector of M contains at most r non-zero coordinates. If $k \geq 1$ is an integer and $n \geq r(k-1)+1$, then in time $|M|^{O(1)}$, we can find a subset K of M of k vectors such that K is M-sum-free.*

Proof. Since the case of $k = 1$ is trivial, we may assume that $k \geq 2$. Let $\mathbf{1} = (1, \ldots, 1)$ be the vector in $\mathbb{F}_2^n$ in which every coordinate is 1. Note that $\mathbf{1} \notin M$. By our assumption M contains a basis B of $\mathbb{F}_2^n$ and we may write $\mathbf{1}$ as a sum of some vectors of B. This implies that $\mathbf{1}$ can be expressed as follows: $\mathbf{1} = v_1 + v_2 + \cdots + v_s$, where $\{v_1, \ldots, v_s\} \subseteq B$ and $v_1, \ldots, v_s$ are linearly independent, and we can find such an expression in polynomial time.

For each $v \in M \backslash \{v_1, \ldots, v_s\}$, consider the set $S_v = \{v, v_1, \ldots, v_s\}$. In polynomial time, we may check whether S_v is linearly independent. Consider two cases:

Case 1: S_v is linearly independent for each $v \in M \backslash \{v_1, \ldots, v_s\}$. Then $\{v_1, \ldots, v_s\}$ is M-sum-free (here we also use the fact that $\{v_1, \ldots, v_s\}$ is linearly independent). Since each v_i has at most r positive coordinates, we have $sr \geq n > r(k-1)$. Hence, $s > k-1$ implying that $s \geq k$. Thus, $\{v_1, \ldots, v_k\}$ is the required set K.

Case 2: S_v is linearly dependent for some $v \in M \backslash \{v_1, \ldots, v_s\}$. Then we can find (in polynomial time) $I \subseteq [s]$ such that $v = \sum_{i \in I} v_i$. Thus, we have a shorter expression for $\mathbf{1}$: $\mathbf{1} = v'_1 + v'_2 + \cdots + v'_{s'}$, where $\{v'_1, \ldots, v'_{s'}\} = \{v\} \cup \{v_i : i \notin I\}$. Note that $\{v'_1, \ldots, v'_{s'}\}$ is linearly independent.

Since $s \leq n$ and Case 2 produces a shorter expression for $\mathbf{1}$, after at most n iterations of Case 2 we will arrive at Case 1. □

Remark 5. *The assumption that M contains a basis is important as otherwise the vectors of M span a subspace of $\mathbb{F}_2^n$ of dimension $n' < n$. Since the subspace is isomorphic to $\mathbb{F}_2^{n'}$, one has to replace n in the theorem by n'.*

Theorem 7. *Let S be an irreducible system and suppose that each equation contains at most r variables. Let $n \geq (k-1)r + 1$ and let $w_{\min}$ be the minimum weight of an equation of S. Then, in time $m^{O(1)}$, we can find an assignment x^0 to variables of S such that $\varepsilon_S(x^0) \geq k \cdot w_{\min}$.*

Remark 6. *To see that the inequality $n \geq r(k-1)+1$ in the theorem is best possible assume that $n = r(k-1)$ and consider a partition of $[n]$ into $k-1$ subsets $N_1, \ldots, N_{k-1}$, each of size r. Let S be the system consisting of subsystems S_p, $p \in [k-1]$, such that a subsystem S_p is comprised of equations $\prod_{i \in I} x_i = -1$ of weight 1 for every I such that $\emptyset \neq I \subseteq N_p$. Now assume without loss of generality that $N_p = [r]$. Observe that the assignment $(x_1, \ldots, x_r) = (1, \ldots, 1)$ falsifies all equations of S_p but by setting $x_j = -1$ for any $j \in [r]$ we satisfy the equation $x_j = -1$ and turn the remaining equations into pairs of the form $\prod_{i \in I} x_i = -1$ and $\prod_{i \in I} x_i = 1$. Thus, the maximum excess of S_p is 1 and the maximum excess of S is $k-1$.*

Remark 7. *It is easy to check that Theorem 7 holds when the weights of equations in S are real numbers, not necessarily integers.*

4.4 Applications of Theorem 7

Theorem 8. *The problem* Max-r-Lin2-AA$[k, r]$ *has a kernel with at most* $(2k-1)r$ *variables.*

Proof. Let T be the system of an instance of Max-r-Lin2-AA$[k, r]$. After applying Rules 4.1 and 4.2 to T as long as possible, we obtain a new system S which is irreducible. Let n be the number of variables in S and observe that the number of variables in an equation in S is bounded by r (as in T). If $n \geq (2k-1)r+1$, then, by Theorem 7 and Remark 1, S is a Yes-instance of Max-r-Lin2-AA$[k, r]$ and, hence, by Lemma 10, S and T are both Yes-instances of Max-r-Lin2-AA$[k, r]$. Otherwise $n \leq (2k-1)r$ and since S is irreducible the number m of equations in S is less than 2^n. Thus, we have the required kernel. □

Corollary 3. *The maximization problem* Max-r-Lin2-AA *is in APX if restricted to* $m = O(n)$ *and the weight of each equation bounded by a fixed constant.*

Proof. It follows from Theorem 7 and Remark 1 that the answer to Max-r-Lin2-AA, as a decision problem, is Yes as long as $2k \leq \lfloor (n+r-1)/r \rfloor$. This implies approximation ratio at most $W/(2\lfloor (n+r-1)/r \rfloor)$ which is bounded by a constant provided $m = O(n)$ and the weight of each equation is bounded by a constant (then $W = O(n)$). □

The (parameterized) Boolean Max-r-Constraint Satisfaction Problem (Max-r-CSP) generalizes MaxLin2-AA$[k, r]$ as follows: We are given a set Φ of Boolean functions, each involving at most r variables, and a collection $\mathcal{F}$ of m Boolean functions, each $f \in \mathcal{F}$ being a member of Φ, and each acting on some subset of the n Boolean variables $x_1, x_2, \ldots, x_n$ (each $x_i \in \{-1, 1\}$). We are to decide whether there is a truth assignment to the n variables such that the total number of satisfied functions is at least $E + k$, where E is the average value of the number of satisfied functions. The parameters are k and r.

Using a bikernelization algorithm described in [3, 25] and our new kernel result, it easy to see that Max-r-CSP with parameters k and r admits a bikernel with at most $(k2^{r+1}-1)r$ variables. This result improves the corresponding result of Kim and Williams [70] ($n \leq kr(r+1)2^r$).

The following result is essentially a corollary of Theorem 7 and Remark 7.

Theorem 9. *Let*

$$f(x) = \hat{f}(\emptyset) + \sum_{I \in \mathcal{F}} \hat{f}(I) \prod_{i \in I} x_i \tag{4.1}$$

be a pseudo-boolean function of degree r*. Then*

$$\max_x f(x) \geq \hat{f}(\emptyset) + \lfloor (\mathrm{rank}A + r - 1)/r \rfloor \cdot \min\{|\hat{f}(I)| : I \in \mathcal{F}\}, \tag{4.2}$$

where A *is a* $(0,1)$*-matrix with entries* a_{ij} *such that* $a_{ij} = 1$ *if and only if term* j *in (4.1) contains* x_i *and* rank$A$ *is the rank of* A *over* $\mathbb{F}_2$*. One can find an assignment of values to* x *satisfying (4.2) in time* $(n|\mathcal{F}|)^{O(1)}$*.*

Proof. By Remark 2 the function $f(x) - \hat{f}(\emptyset) = \sum_{I \in \mathcal{F}} \hat{f}(I) \prod_{i \in I} x_i$ is the excess over the system $\prod_{i \in I} x_i = b_I$, $I \in \mathcal{F}$, where $b_I = +1$ if $\hat{f}(I) > 0$ and $b_I = -1$ if $\hat{f}(I) < 0$, with weights $|\hat{f}(I)|$. Clearly, Rule 4.1 will not change the system. Using Rule 4.2 we can replace the system by an equivalent one (by Lemma 10) with rank$A$ variables. By Lemma 11, the new system is irreducible and we can now apply Theorem 7. By this theorem, Remark 2 and Remark 7, $\max_x f(x) \geq \hat{f}(\emptyset) + k^* \min\{|\hat{f}(I)| : I \in \mathcal{F}\}$, where k^* is the maximum value of k satisfying rank$A \geq (k-1)r + 1$. It remains to observe that $k^* = \lfloor (\text{rank}A + r - 1)/r \rfloor$. □

To give a new proof of the Edwards-Erdős bound, we need the following well-known and easy-to-prove lemma [13]. For a graph $G = (V, E)$, an incidence matrix is a $(0,1)$-matrix with entries $m_{e,v}$, $e \in E$, $v \in V$ such that $m_{e,v} = 1$ if and only if v is incident to e.

Lemma 15. *The rank over $\mathbb{F}_2$ of an incidence matrix M of a connected graph equals $|V| - 1$.*

Theorem 10. *Let $G = (V, E)$ be a connected graph with n vertices and m edges. Then G contains a bipartite subgraph with at least $\frac{m}{2} + \frac{n-1}{4}$ edges. Such a subgraph can be found in polynomial time.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ and let $c : V \to \{-1, 1\}$ be a 2-coloring of G. Observe that the maximum number of edges in a bipartite subgraph of G equals the maximum number of properly colored edges (i.e., edges whose end-vertices received different colors) over all 2-colorings of G. For an edge $e = v_i v_j \in E$ consider the following function $f_e(x) = \frac{1}{2}(1 - x_i x_j)$, where $x_i = c(v_i)$ and $x_j = c(v_j)$ and observe that $f_e(x) = 1$ if e is properly colored by c and $f_e(x) = 0$, otherwise. Thus, $f(x) = \sum_{e \in E} f_e(x)$ is the number of properly colored edges for c. We have $f(x) = \frac{m}{2} - \frac{1}{2} \sum_{e \in E} x_i x_j$. By Theorem 9, $\max_x f(x) \geq m/2 + \lfloor (\text{rank}A + 2 - 1)/2 \rfloor / 2$. Observe that matrix A in this bound is an incidence matrix of G and, thus, by Lemma 15 rank$A = n - 1$. Hence, $\max_x f(x) \geq \frac{m}{2} + \frac{1}{2} \lfloor \frac{n}{2} \rfloor \geq \frac{m}{2} + \frac{n-1}{4}$ as required. □

This theorem can be extended to the Balanced Subgraph problem [8], where we are given a graph $G = (V, E)$ in which each edge is labeled either by $=$ or by $\neq$ and we are asked to find a 2-coloring of V such that the maximum number of edges is satisfied; an edge labeled by $=$ ($\neq$, resp.) is *satisfied* if and only if the colors of its end-vertices are the same (different, resp.).

Theorem 11. *Let $G = (V, E)$ be a connected graph with n vertices and m edges labeled by either $=$ or $\neq$. There is a 2-coloring of V that satisfies at least $\frac{m}{2} + \frac{n-1}{4}$ edges of G. Such a 2-coloring can be found in polynomial time.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ and let $c : V \to \{-1, 1\}$ be a 2-coloring of G. Let $x_p = c(v_p)$, $p \in [n]$. For an edge $v_i v_j \in E$ we set $s_{ij} = 1$ if $v_i v_j$ is labeled by $\neq$ and $s_{ij} = -1$ if $v_i v_j$ is labeled by $=$. Then the function $\frac{1}{2} \sum_{v_i v_j \in E} (1 - s_{ij} x_i x_j)$ counts the number of edges satisfied by c. The rest of the proof is similar to that in the previous theorem. □

Chapter 5

MaxLin2 Parameterized Below W

5.1 Introduction

In this chapter we consider the following problem:

MaxLin2-BW

Instance: A system S of m linear equations in n variables over $\mathbb{F}_2$, where no equation has more than $r = r(n)$ variables and equation j is assigned a positive integral weight w_j, $j = 1, \ldots, m$, and a positive real k. We will write equation j in S as $\sum_{i \in \alpha_j} z_i = b_j$, where $\emptyset \neq \alpha_j \subseteq \{1, 2, \ldots, n\}$ and $|\alpha_j| \leq r$.

Parameter: k.

Question: Is there an assignment of values to the n variables such that the total weight of the satisfied equations is at least $W - k$, where $W = w_1 + \cdots + w_m$?

Let Max-($\leq r$,$\leq s$)-Lin2 (Max-($= r$,$= s$)-Lin2, respectively) denote the problem MaxLin2 restricted to instances, which have at most (exactly, respectively) r variables in each equation and at most (exactly) s appearances of any variable in all equations. In the special case when each equation has weight 1 and there are no two equations with the same left-hand side, MaxLin2-BW will be denoted by MaxLin2-B[m]. We will prove that MaxLin2-BW remains hard even after significant restrictions are imposed on it, namely, even Max-($= 3$,$= 3$)-Lin2-B[m] is W[1]-hard. This is proved in Section 5.2.

No further improvement of this result is possible unless FPT=W[1] as we will prove that Max-(≤ 2,*)-Lin2-BW is fixed-parameter tractable, where symbol * indicates that no restriction is imposed on the number of appearances of a variable in the equations. Moreover, we will show that the nonparameterized problem Max-(*,≤ 2)-Lin2 is polynomial time solvable, where symbol * indicates that no restriction is imposed on the number of variables in any equation. These two

results are shown in Section 5.3.

5.2 Hardness Results

In the problem Odd Set, given a set $V = \{1, 2, \ldots, n\}$, distinct sets $e_1, e_2, \ldots, e_m \subseteq V$ and a nonnegative integer k, we are to decide whether we can pick a set R of at most k elements in V such that R intersects all sets e_i in an odd number of elements. Downey *et al.* [34] showed the problem is $W[1]$-hard by a reduction from Perfect Code.

We prove that Max-(= 3,= 3)-Lin2-B[m] is $W[1]$-hard in two parts. First, we give a reduction from Odd Set to show Max-($\leq$ 3,*)-Lin2-BW is $W[1]$-hard. Then, we give a reduction from Max-($\leq$ 3,*)-Lin2-BW to Max-(= 3,= 3)-Lin2-B[m].

Lemma 16. *The problem* Max-($\leq$ 3,*)-Lin2-BW *is* $W[1]$*-hard.*

Proof. Consider an instance of Odd Set with elements $1, 2, \ldots, n$ and distinct sets $e_1, e_2, \ldots, e_m$, with parameter k.

Create an instance of Max-($\leq$ 3,*)-Lin2-BW with parameter k as follows. Start with the variables $x_1, x_2, \ldots, x_n$ and equations $x_1 = 0, x_2 = 0, \ldots, x_n = 0$ (each of weight 1). For every set $e_i = \{j_1, j_2, j_3, \ldots, j_{n_i}\}$ do the following. Add the variables $y^i_1, \ldots, y^i_{n_i-1}$ and the following set E_i of equations, each of weight $k + 1$.

$$
\begin{aligned}
&y^i_1 + x_{j_1} = 0\\
&y^i_1 + y^i_2 + x_{j_2} = 0\\
&y^i_2 + y^i_3 + x_{j_3} = 0\\
&\ldots\\
&y^i_{n_i-2} + y^i_{n_i-1} + x_{j_{n_i-1}} = 0\\
&y^i_{n_i-1} + x_{j_{n_i}} = 1
\end{aligned}
$$

Observe that the number of variables and equations is polynomial in nm. It remains to show that this instance of Max-($\leq$ 3,*)-Lin2-BW is a Yes-instance if and only if the instance of Odd Set is a Yes-instance.

Suppose first that we can satisfy simultaneously equations of total weight at least $W - k$. Consider the set E_i of equations. Since the equations have weight $k+1$, they must all be satisfied. By summing them up, we obtain $\sum_{a=1}^{n_i} x_{j_a} = 1$ over $\mathbb{F}_2$. Therefore an odd number of the values of $x_{j_1}, x_{j_2}, \ldots, x_{j_{n_i}}$ are 1. Note that at most k equations of the type $x_i = 0$ are not satisfied. The above implies that if R is the set of elements, j, for which the corresponding variable, x_j is equal to 1, then the size of R is at most k and for each set e_i the intersection of R and e_i is odd. Therefore R has the desired property.

Conversely, suppose R is a set of at most k elements such that for each set e_i the intersection of R and e_i is odd. Set $x_j = 1$ if and only if j is in R. Also set $y^i_1 = x_{j_1}$ and $y^i_{r+1} = y^i_r + x_{j_{r+1}}$ for $1 \leq r \leq n_i - 2$. It now follows immediately that all equations of E_i, apart from the last one, are satisfied. The last one is satisfied as well because by summing up all the equations, we obtain

$\sum_{a=1}^{n_i} x_{j_a} = 1$, which is satisfied. Since $|R| \leq k$, we will satisfy simultaneously equations of total weight at least $W - k$. □

Theorem 12. *The problem* Max-(= 3,= 3)-Lin2-B⌈m⌋ *is* $W[1]$*-hard.*

Proof. Observe that in the system obtained in the proof of Lemma 16 no two equations have the same left-hand side. Consider an instance of Max-($\leq$ 3,*)-Lin2-BW in which no two equations have the same left-hand side. Hereafter, we view a single equation of weight w as w identical equations of weight 1. This means we do have equations with the same left-hand side (for $k \geq 1$), but note that these equations also have the same right-hand side.

For each of the reductions that follow, we show that the optimal assignment will falsify the same number of equations in the original instance as in the reduced instance. This implies that the original instance is a Yes-instance if and only if the reduced instance is a Yes-instance.

For each variable x, let $d(x)$ denote the total number of equations containing x. We first apply the following two reduction rules until $d(x) \leq 3$ for every variable x.

If $d(x) = 4$, replace x with four new variables, x_1, x_2, x_3, x_4. For each equation containing x, replace the occurrence of x with one of x_1, x_2, x_3, x_4, so that each new variable appears once. Furthermore, add equations $x_1 + x_2 = 0$, $x_2 + x_3 = 0$, $x_3 + x_4 = 0$, $x_4 + x_1 = 0$.

If $d(x) \geq 5$, replace x with six new variables, $x_1, x_2, x_3, x_4, x_5, x_6$. For each equation containing x, replace the occurrence of x with one of $x_1, x_2, x_3, x_4, x_5, x_6$, distributing the new x_i evenly among the equations, so that

$$\lfloor d(x)/6 \rfloor \leq d(x_1) \leq d(x_2) \leq d(x_3) \leq d(x_4) \leq d(x_5) \leq d(x_6) \leq \lceil d(x)/6 \rceil.$$

Furthermore, add $\lceil (d(x)-2)/6 \rceil$ copies of each of the equations $x_1+x_2 = 0$, $x_2+x_3 = 0$, $x_3+x_4 = 0$, $x_4 + x_5 = 0$, $x_5 + x_6 = 0$, $x_6 + x_1 = 0$, $x_1 + x_4 = 0$, $x_2 + x_5 = 0$, $x_3 + x_6 = 0$.

Observe that each rule replaces a variable with a set of variables, each of which appears in fewer equations than the original variable. Therefore after enough applications, each variable will appear in at most three equations. To see that only a polynomial number of applications are needed, observe that at each iteration $\max_i d(x_i) \leq 8d(x)/9$. Therefore we may view the applications of reduction rules as a branching tree for each variable, where the depth of the tree for a variable x is bounded by $\log_{9/8} d(x)$ and the tree branches at most six ways each time.

We now show that each rule is valid.

For the $d(x) = 4$ case, suppose that the optimal assignment is one in which x_1, x_2, x_3, x_4 are not all the same. Then at least two of the equations $x_1 + x_2 = 0$, $x_2 + x_3 = 0$, $x_3 + x_4 = 0$, $x_4 + x_1 = 0$ will be falsified, but then we can satisfy all of them by falsifying at most two other equations. Hence, there exists an optimal assignment in which x_1, x_2, x_3, x_4 all have the same value.

For the $d(x) \geq 5$ case, suppose that the optimal assignment is one in which the new variables $x_1, x_2, x_3, x_4, x_5, x_6$ are not all the same. Consider the set of nine equations $x_1+x_2 = 0$, $x_2+x_3 = 0$, $x_3 + x_4 = 0$, $x_4 + x_5 = 0$, $x_5 + x_6 = 0$, $x_6 + x_1 = 0$, $x_1 + x_4 = 0$, $x_2 + x_5 = 0$, $x_3 + x_6 = 0$.

If the value of exactly one new variable is different from the values of the rest of the variables, at least three of the nine equations will be falsified. In the worst case changing the value of this variable will falsify at most $d(x_6) \leq \lceil d(x)/6 \rceil \leq 3\lceil (d(x)-2)/6 \rceil$ equations. If the values of exactly two new variables are different from the values of the rest of the variables, at least four of the nine equations will be falsified. Changing the value of the two variables will falsify at most $d(x_5) + d(x_6) \leq 4\lceil (d(x)-2)/6 \rceil$ equations. If three new variables are assigned one, and three are assigned zero, at least five of the nine equations will be falsified. Assigning zero to all new variables will falsify at most $d(x_4) + d(x_5) + d(x_6) \leq 5\lceil (d(x)-2)/6 \rceil$ equations. (For example, if $d(x) = 8$ then $d(x_4) = 1$, $d(x_5) = d(x_6) = 2$ and $\lceil (d(x)-2)/6 \rceil = 5$.) Therefore, there exists an optimal assignment in which $x_1, x_2, x_3, x_4, x_5, x_6$ all have the same value.

Thus, for each rule the new instance has an optimal assignment in which all the new equations are satisfied, and all the new variables have the same value. By setting x to this value, we have an assignment to the original instance that falsifies the same number of equations. Consider now an optimal assignment to the original instance and produce an assignment to the new variables which gives each new variable the same value as x. As any of new equations has exactly two variables, the number of equations falsified in the new instance is the same as in the original one.

We now have an instance in which each equation contains at most three variables and each variable appears in at most three equations. Next, we observe that we may map this to an instance where each equation contains exactly three variables.

First consider equations containing one variable. If an equation is of the form $x = 0$, this may be replaced with equations $a + b + x = 0$, $u + v + a = 0$, $u + v + b = 0$, where a, b, u, v are new variables. For $x = 1$, we replace it with the equations $a + b + x = 1$, $u + v + a = 0$, $u + v + b = 0$.

For equations containing two variables, if an equation is of the form $x + y = 0$, this may be replaced with equations $u + v + x = 0$ and $u + v + y = 0$, where u, v are new variables. A similar mapping may be done for $x + y = 1$ by replacing y with $y + 1$.

Observe that for each reduction, if an assignment to the original instance satisfies the original equation, it can be extended to one that satisfies all the new equations, and if it does not, an optimal extension will satisfy all but one of the new equations. Thus an optimal assignment to the original instance falsifies the same number of equations as an optimal assignment to the reduced instance.

We now have that every equation contains exactly three variables and each variable is in at most three equations. We now map this to an instance where every variable is in exactly three equations.

If a variable x only appears in one equation, then we may assume that this equation is satisfied, and remove it from the system. Since each equation contains exactly 3 variables, the number of variables x with $d(x) = 2$ must be a multiple of 3. Thus, we may partition the variables x with $d(x) = 2$ into triplets.

Consider each triplet x_1, x_2, x_3 such that $d(x_1) = d(x_2) = d(x_3) = 2$. Add variables z_1, z_2, z_3, $u_1, \ldots, u_6$, and equations $x_1 + x_2 + u_1 = 0$, $u_1 + u_2 + z_1 = 0$, $u_2 + u_3 + z_1 = 0$ (two copies),

$u_3 + u_1 + z_2 = 0$, $x_3 + u_4 + z_2 = 0$, $u_4 + u_5 + z_2 = 0$, $u_5 + u_6 + z_3 = 0$ (two copies), $u_6 + u_4 + z_3 = 0$.

Observe that for any assignment to x_1, x_2, x_3, it is possible to satisfy all these equations by setting $z_1 = z_2 = z_3 = 0$, $u_1 = u_2 = u_3 = x_1 + x_2$ and $u_4 = u_5 = u_6 = x_3$. Thus an optimal assignment to the original instance extends to an optimal assignment to the reduced instance that falsifies the same number of equations.

We now have that every equation has exactly three variables and every variable appears in exactly three equations. It remains to show that we can map this to an instance in which these properties hold and no two equations have the same left-hand side. Since we started the proof of this theorem from a system where every pair of equations with the same left-hand side had the same right-hand side, and since our transformations above have not changed this property, it suffices to get rid of identical equations.

Note that since $d(x) = 3$ for every variable x, there at most three copies of any given equation in the system.

If there are three copies of the same equation, then none of the variables appearing in that equation appear anywhere else. Therefore we may assume the equation is satisfied, and remove the three copies from the system.

If there are two copies of the equation $x + y + z = 0$, replace them with the following set of six equations: $x + y + c_1 = 0$, $a_1 + b_1 + c_1 = 0$, $a_1 + b_1 + z = 0$, $x + y + c_2 = 0$, $a_2 + b_2 + c_2 = 0$, $a_2 + b_2 + z = 0$, $a_1 + b_2 + c_1 = 0$, $a_2 + b_1 + c_2 = 0$, where $a_1, b_1, c_1, a_2, b_2, c_2$ are new variables. Observe that if $x + y + z = 0$ is satisfied then by setting $a_1 = a_2 = x$, $b_1 = b_2 = y$, $c_1 = c_2 = z$, we can satisfy all of these equations. If $x + y + z = 0$ is falsified, then the first three equations ($x + y + c_1 = 0$, $a_1 + b_1 + c_1 = 0$, $a_1 + b_1 + z = 0$) are inconsistent, as are $x + y + c_2 = 0$, $a_2 + b_2 + c_2 = 0$, $a_2 + b_2 + z = 0$, and hence at least two equations of the set of six equations are falsified. Furthermore, by setting $a_1 = a_2 = b_1 = b_2 = c_1 = c_2 = 0$ we can satisfy all but two equations in the set of six equations. Thus in either case, an optimal assignment to the original instance extends to an optimal assignment to the reduced instance that falsifies the same number of equations.

If there are two equations of the form $x + y + z = 1$, do the same as above but change all the right-hand sides to 1. □

5.3 Algorithmic Results

In this section, we assume that all weights of the equations in MaxLin2-BW belong to the set $\{1, 2, \ldots, k + 1\}$. Indeed, replacing any weight larger than $k + 1$ by $k + 1$ does not change the answer to MaxLin2-BW.

In the Edge Bipartization problem, given a graph G and a nonnegative integer k, we are to decide whether we can make G bipartite by deleting at most k edges. When k is the parameter, the problem is fixed-parameter tractable and can be solved by an algorithm of running time $O(2^k M^2)$

[58], where M is the number of edges in G. We prove that MAX-($\leq$ 2,*)-LIN2-BW is fixed-parameter tractable by giving a reduction to EDGE BIPARTIZATION.

Theorem 13. *The problem* MAX-($\leq$ 2,*)-LIN2-BW *can be solved in time* $O(2^k(km)^2)$.

Proof. Consider an instance S of MAX-($\leq$ 2,*)-LIN2-BW with m equations and consider an assignment which minimizes the weight of falsified equations. Now replace every equation, $x_i+x_j = 0$, which contains two variables in the left-hand side and 0 in the right-hand side, by two equations $x_i + y = 1$ and $x_j + y = 1$, where y is a new variable; both equations have the same weight as $x_i + x_j = 0$. If the assignment satisfies $x_i + x_j = 0$ then both new equations can be satisfied by extending the assignment with $y = 1 - x_i$. If the assignment falsifies $x_i + x_j = 0$ then exactly one of the two new equations will be falsified by extending the assignment with $y = 0$. Thus, the replacement preserves the minimum weight of falsified equations and can be used to replace the instance by an equivalent one S' in which every equation with two variables has right-hand side equal 1.

Now assume that all equations of the system with two variables have right-hand side equal 1 and construct the following weighted graph G. The vertices of G are the variables of the system plus two extra vertices, v' and v''. For each equation $x + y = 1$ in S', add to G edge xy. For every equation $x = b$ of S', add to G edge $v'x$ if $b = 0$ and xv'' if $b = 1$. The weight of each edge coincides with the weight of the corresponding equation of S'. Finally, add to G edge $v'v''$ of weight $k + 1$.

Let w^* be a nonnegative integer such that $w^* \leq k$. Observe that there is an assignment which falsifies equations of total weight w^* if and only if there is a set F of edges of G of total weight w^* such that $G - F$ is bipartite. Indeed, consider an assignment which falsifies equations of total weight w^*. Initiate sets V' and V'' as follows: $V' = \{v'\}$ and $V'' = \{v''\}$. If an equation $u + v = 1$ is satisfied and $u = 1,\ v = 0$, then u is added to V' and v to V'', and if an equation $x = b$ is satisfied, then x is added to V' if $b = 1$ and to V'' if $b = 0$. Note that the total weight of edges whose both end-vertices are either in V' or in V'' equals w^*. Similarly, we can show the other direction.

To get rid of the weights in G we replace each edge uv of G of weight w with w paths $uy^p_{uv}z^p_{uv}v$, $p \in \{1, \ldots, w\}$, where y^p_{uv} and z^p_{uv} are new vertices. This finally reduces the instance of MAX-($\leq$ 2,*)-LIN2-BW into an instance of EDGE BIPARTIZATION with $O(mk)$ edges. It remains to apply the EDGE BIPARTIZATION algorithm mentioned before the theorem. □

Theorem 14. *The problem* MAX-(*,$\leq$ 2)-LIN2 *is polynomial time solvable.*

Proof. Consider an instance of MAX-(*,$\leq$ 2)-LIN2 with system $Ax = b$ in which each equation has a weight. We start by applying the following reduction rule as long as possible: if there is a variable which appears only in one equation, delete the equation from the system. Since we can always satisfy an equation with a unique variable, the reduction rule produces a new system $A'x' = b'$ such that the minimum weight of equations falsified by an assignment is the same in $Ax = b$ as in $A'x' = b'$. Now construct a graph G whose vertices correspond to equations in $A'x' = b'$ and a pair of vertices is adjacent if the corresponding equations share a variable.

Consider a connected component H in G and the subsystem $A''x'' = b''$ corresponding to H. Let $b'' = (b''_1, \ldots, b''_{m''})$, where m'' is the number of rows in A''. Recall that $A''x'' = b''$ is a system over $\mathbb{F}_2$, and thus all summations and ranks of matrices considered below are over $\mathbb{F}_2$. Since the sum of rows in A'' is equal 0 and any subsum of the sum is not equal 0, the rank of A'' equals $m'' - 1$. Observe that if $\sum_{j=1}^{m''} b'' = 0$ then the rank of the matrix $[A''b'']$ equals the rank of A'' and, thus, there is an assignment which satisfies all equations in $A''x'' = b''$. However, if $\sum_{j=1}^{m''} b'' = 1$, the rank of $[A''b'']$ is m'' but the rank of A'' is $m'' - 1$. Hence, the system $A''x'' = b''$ is no longer consistent and we can satisfy all equations but one. The falsified equation can be chosen arbitrarily and to maximize the total weight of satisfied equations of $A''x'' = b''$ we have to choose an equation of minimum weight.

The above argument leads to a polynomial time algorithm to solve Max-(*,≤ 2)-Lin2. □

Remark 8. We can prove Theorem 14 using another approach, whose idea we will briefly describe. Consider a pair of equations from the system $Ax = b$ which share a variable x_i and consider an assignment which minimizes the weight of falsified equations. Let w^* be the smallest weight of the two equations. At least one of the two equations is satisfied by the assignment and if one of the two equations is falsified, its weight is w^* (as otherwise we could change the value of x_i, arriving at a contradiction). Replace the two equations in $Ax = b$ by the equation which is the sum of these two equations and whose weight equals w^*. Observe that the replacement does not change the minimum total weight of falsified equations. Theorem 14 can be proved by repeatedly using this reduction.

Part II

Parameterized Complexity of MaxSAT

Chapter 6

Parameterized Complexity of MaxSAT Above Average

6.1 Introduction

In this chapter we consider the problem of MaxSat-AA:

MaxSat-AA
Instance: A CNF formula F with clauses $c_1, \ldots, c_m$, and variables $x_1, \ldots, x_n$, and a nonnegative integer k. Clause c_i has r_i literals, $i = 1, \ldots, m$.
Parameter: k.
Task: Decide whether there is a truth assignment satisfying at least $\text{asat}(F)+k$ clauses, where $\text{asat}(F) = \sum_{i=1}^{m}(1 - 2^{-r_i})$.

The problem Max-$r(n)$Sat-AA is a refinement of MaxSat-AA in which each clause has at most $r(n)$ literals, where $r(n)$ is an integer-valued function.

Note that $\text{asat}(F)$ is the average number of satisfied clauses. Indeed, if we assign TRUE or FALSE to each x_j with probability $1/2$ independently of the other variables, then the probability of c_i being satisfied is $1 - 2^{-r_i}$, and by linearity of expectation, $\text{asat}(F)$ is the expected number of satisfied clauses. (Since our distribution is uniform, $\text{asat}(F)$ is indeed the average number of satisfied clauses.) Let $\text{sat}(F)$ denote the maximum number of clauses satisfied by a truth assignment. For Boolean variables $y_1, \ldots, y_t$, the *complete set of clauses on* $y_1, \ldots, y_t$ is the set $\{(z_1 \vee \ldots \vee z_t) : z_i \in \{y_i, \overline{y}_i\}, i \in [t]\}$. Any formula F consisting of one or more complete sets of clauses shows that the lower bound $\text{sat}(F) \geq \text{asat}(F)$ on $\text{sat}(F)$ is tight. Using the derandomization method of conditional expectations (see, e.g., Alon and Spencer [5]), it is easy to obtain a polynomial time algorithm

which finds a truth assignment satisfying at least asat(F) clauses. Thus, the question asked in MaxSat-AA is whether we can find a better truth assignment efficiently from the parameterized complexity point of view.

6.2 Hardness Results

In this section we give our hardness results. A *3-CNF formula* contains exactly three literals in each clause. For our results we need the following problem as a starting point for our reductions. Here c is a positive integral constant.

Linear-c-3Sat
Instance: A 3-CNF formula F with clauses $c_1, \ldots, c_m$, and variables $x_1, \ldots, x_n$ such that $m \leq cn$ (i.e., the number of clauses in F is linear in the number of variables).
Task: Decide whether there is a truth assignment satisfying F.

It is well known that Linear-c-3Sat is NP-complete for each $c \geq 2$: the well-known theorem of Tovey [104] states that the 3-SAT problem is NP-complete even when the input consists of 3-CNF formula with every variable appearing in at most four clauses.

We begin with a useful construction of a Max-$r(n)$Sat-AA instance from an arbitrary Linear-c-3Sat instance.

Lemma 17. *Let c be a positive integral constant. For a* Linear-c-3Sat *instance F with n variables and m clauses, we can construct, in polynomial time, a CNF formula F' with $n' = 2cn$ variables, such that each clause of F' has $r(n') = \lceil \log n' \rceil$ literals, and F is satisfiable if and only if (F',2) is a* Yes*-instance of* Max-$r(n')$-Sat-AA.

Proof. Consider a Linear-c-3Sat instance F. Let the variables of F be $x_1, \ldots, x_n$, and the distinct clauses $c_1, \ldots, c_m$. Recall that $m \leq cn$.

We form a CNF formula F' with $n' = 2cn$ variables, the existing variables $x_1, \ldots, x_n$, together with new variables $y_1, \ldots, y_{n'-n}$, and $m' = 2^{\lceil \log n' \rceil + 1}$ clauses. The set of clauses of F' consists of three sets, C_1, C_2 and C_3, described below:

- C_1 is the complete set of clauses on variables $y_1, \ldots, y_{\lceil \log n' \rceil}$ without the clause consisting of all negative literals, $\overline{c} = (\overline{y}_1 \vee \overline{y}_2 \vee \ldots \vee \overline{y}_{\lceil \log n' \rceil})$.
- $C_2 = \{c_i \vee \overline{y}_4 \vee \overline{y}_5 \vee \ldots \vee \overline{y}_{\lceil \log n' \rceil} : \ i \in [m]\}$.
- C_3 is a set of $m' - |C_1| - |C_2|$ clauses on the variables $y_{\lceil \log n' \rceil + 1}, \ldots, y_{n'-n}$, of length $\lceil \log n' \rceil$ such that each variable appears in at least one clause and every clause consists of only positive literals.

We claim that F is satisfiable if and only if $(F', 2)$ is a YES-instance of MAX-$r(n)$SAT-AA, thus completing the proof. Since, in F', each clause has $r(n') = \lceil \log n' \rceil$ literals, we have $\text{asat}(F') = (1 - 2/m')m' = m' - 2$. Thus, $(F', 2)$ asks whether all the clauses of F' can be satisfied.

Suppose F is satisfied by a truth assignment. Extend this assignment to the variables of F' by assigning all y_i to be TRUE. Since F is satisfied, all the clauses in C_2 are satisfied. Every clause in C_1 and C_3 contains at least one positive literal y_i, and so is satisfied. Hence F' is satisfied.

If F' is satisfied, then $y_1, \ldots, y_{\lceil \log n' \rceil}$ must all be set to TRUE (otherwise, there is a clause in C_1 that is not satisfied). As a result, the set C_2 of clauses can be simplified to the 3-SAT formula F, and thus F must be satisfied. □

Lemma 17 implies the following:

Theorem 15. MAX-$r(n)$SAT-AA *is para-NP-complete for any* $r(n) \geq \lceil \log n \rceil$.

Proof. Since it suffices to prove the theorem for $r(n) = \lceil \log n \rceil$ we assume that $r(n) = \lceil \log n \rceil$. MAX-$r(n)$SAT-AA is in para-NP as in polynomial time, we can calculate $\text{asat}(\varphi) + k$, and then decide if there exists an assignment satisfying this many clauses in polynomial time using a non-deterministic Turing machine. To complete our proof of para-NP-completeness, we observe that Lemma 17 gives a reduction from LINEAR-c-3SAT to MAX-$r(n)$SAT-AA with $k = 2$. □

It is not hard to prove Theorem 15 without starting from LINEAR-c-3SAT. We use LINEAR-c-3SAT to ensure that $n' = O(n)$ which is necessary in the proof of the next theorem. Hereafter, we will assume the Exponential Time Hypothesis (ETH), which is stated below.

Exponential Time Hypothesis (ETH) (Impagliazzo and Paturi [65])**.** *There is a positive real s such that 3-SAT on n variables cannot be solved in time $O(2^{sn})$.*

Using the Sparsification Lemma [66, Corollary 1], one may assume that, in the ETH in the input formula F to 3-SAT, the number of clauses is at most p times the number of variables, for some positive constant p. For completeness we give a proof below.

Sparsification Lemma (Impagliazzo *et al.* [66])**.** *For every $\epsilon > 0$, there is a constant C_ϵ so that any 3-CNF formula F with n variables, can be expressed as $F = \vee_{i=1}^{t} Y_i$, where $t \leq 2^{\epsilon n}$ and each Y_i is a 3-CNF formula with all variables from F and with at most $C_\epsilon n$ clauses. Moreover, this disjunction can be computed by an algorithm running in time $O(2^{\epsilon n})$.*

Proposition 1 ([66])**.** *Assuming the ETH, there is a positive integral constant C and a positive real s' such that* LINEAR-C-3SAT *cannot be solved in time $O(2^{s'n})$, where n is the number of variables.*

Proof. Let s be a positive real such that 3-SAT on n variables cannot be solved in time $O(2^{sn})$, let $s' = s/2$ and let $C = C_{s'}$ be a constant sufficient for the Sparsification Lemma (where $\epsilon = s'$).

Suppose that LINEAR-C-3SAT can be solved in time $O(2^{s'n})$. Using the Sparsification Lemma, we can produce 3-CNF formulas $Y_1, \ldots, Y_t, t \leq 2^{s'n}$, in time $O(2^{s'n})$. We can solve all of them in time $O(2^{2s'n}) = O(2^{sn})$ and so can obtain a solution for F in time $O(2^{sn})$, a contradiction. □

Thus, it follows that there exists a constant C such that Linear-C-3SAT cannot be solved in time $2^{o(n)}$ unless the ETH fails. Using the ETH, we strengthen Theorem 15.

Theorem 16. *Assuming the ETH,* Max-$r(n)$Sat-AA *is not in XP for any* $r(n) \geq \log\log n + \varphi(n)$*, where* $\varphi(n)$ *is any real-valued unbounded strictly increasing computable function of* n*.*

Proof. Let $\varphi(n)$ be a real-valued unbounded strictly increasing function of n. Note that if the theorem holds for some unbounded strictly increasing function $\psi(n)$ of n, it holds also for any strictly increasing function $\psi'(n)$ such that $\psi'(n) \geq \psi(n)$ for every $n \geq 1$. Thus, it suffices to give the proof for $\varphi(n) \leq \log\log n$.

Let C be a constant from Proposition 1. Consider a Linear-C-3SAT instance F'' with n'' variables and m'' clauses. Using Lemma 17, reduce F'' to a Max-$r'(n')$-Sat-AA instance (F', k) with n' variables and $r'(n') = \lceil \log n' \rceil$. Note that $n' = 2Cn''$.

Now let n be the minimum integer such that $\lceil \log n' \rceil \leq \log\log n + \varphi(n)$. Add $n - n'$ new variables to F' together with a pair of contradicting unit clauses, (x), $(\overline{x})$ for each new variable. Let F denote the resulting formula and let (F, k) be an instance of Max-$r(n)$-Sat-AA. (The extra variables can be seen as padding, so that $r(n)$ has the required value.) The total number n of variables in F is such that $r(n) = \lceil \log n' \rceil \leq \log\log n + \varphi(n)$. By the minimality of n, we have $1 + \log n' > \log\log(n-1) + \varphi(n-1)$ and, thus, $n - 1 \leq 2^{2n'/2^{\varphi(n-1)}} \leq 2^{2n'/2^{\varphi(n')}} = 2^{o(n')}$. Hence, it takes $2^{o(n')}$ time to construct F, the last equality holding because $\varphi(n)$ is strictly increasing. Observe that F has $O(n)$ clauses.

Note that (F', k) is a Yes-instance of Max-$r(n')$-Sat-AA if and only if (F, k) is a Yes-instance of Max-$r(n)$-Sat-AA. By Lemma 17, $(F', 2)$ is a Yes-instance of Max-$r(n')$-Sat-AA if and only if the Linear-C-3SAT instance F'' is satisfiable. Thus, $(F, 2)$ is a Yes-instance of Max-$r(n)$-Sat-AA if and only if F'' is satisfiable. Therefore, if there was an XP algorithm for Max-$r(n)$Sat-AA, then for $k = 2$ it would have running time $O(n^c) = 2^{o(n')} = 2^{o(n'')}$, where c is a constant, to check whether the Linear-C-3SAT instance F'' is satisfiable, contradicting Proposition 1. □

6.3 Algorithmic Results

To prove the main result of this section, Theorem 18, we reduce Max-$r(n)$Sat-AA to Max-$r(n)$Lin2-AA considered in the previous chapter. Recall the problem definition:

Max-$r(n)$Lin2-AA

Instance: A system S of equations $\prod_{i \in I_j} x_i = b_j$, where $b_j \in \{-1, 1\}$, $|I_j| \leq r(n)$, $x_i \in \{-1, 1\}$, and $j \in [m]$, in which Equation j is assigned a positive integral weight w_j, and a nonnegative integer k

Parameter: k.

Task: Decide whether $\mathrm{sat}(S) \geq W/2 + k$, where $W = \sum_{j=1}^{m} w_j$.

If $r(n) = n$, MAX-$r(n)$LIN2-AA will be denoted MAXLIN2-AA.

The *excess* for $x = (x_1, \ldots, x_n) \in \{-1, 1\}^n$ of S is

$$\varepsilon_S(x) = \frac{1}{2}\left[\sum_{j=1}^{m} c_j \prod_{i \in I_j} x_i\right],$$

where $c_j = w_j b_j$. Observe that $\varepsilon_S(x)$ is the difference between the total weight of equations satisfied by x and $W/2$. The *maximum excess* of S is $\max\{\varepsilon_S(x) : x \in \{-1, 1\}^n\}$. Thus, the answer to MAXLIN2-AA is YES if and only if the maximum excess of S is at least k.

Consider two reduction rules for MAXLIN2 stated in the previous chapter, and first introduced in [49]:

Reduction Rule 4.1. *If we have, for a subset I of $[n]$, an equation $\prod_{i \in I} x_i = b'_I$ with weight w'_I, and an equation $\prod_{i \in I} x_i = b''_I$ with weight w''_I, then we replace this pair by one of these equations with weight $w'_I + w''_I$ if $b'_I = b''_I$ and, otherwise, by the equation whose weight is bigger, modifying its new weight to be the difference of the two old ones. If the resulting weight is 0, we delete the equation from the system.*

Reduction Rule 4.2. *Let A be the matrix over $\mathbb{F}_2$ corresponding to the set of equations in S, such that $a_{ji} = 1$ if variable x_i appears in Equation j, and 0 otherwise. Let $t = \mathrm{rank}A$ and suppose columns $a^{i_1}, \ldots, a^{i_t}$ of A are linearly independent. Then delete all variables not in $\{x_{i_1}, \ldots, x_{i_t}\}$ from the equations of S.*

The two reduction rules are of interest due to the following:

Lemma 18. *[49] Let S' be obtained from S by 4.1 or Rule 4.2. Then the maximum excess of S' is equal to the maximum excess of S. Moreover, S' can be obtained from S in time polynomial in n and m.*

Recall In Chapter 4 we showed the following:

Theorem 17. *Let (S, k) be an instance of* MAXLIN2-AA *such that system S cannot be reduced by Rules 4.2 and 4.1. Let n be the number of variables in S and let r be the maximum number of variables in an equation of S. If $n \geq (2k-1)r + 1$ then the answer to (S, k) is* YES.

Let (F, k) be an instance of MAXSAT-AA given by a CNF formula F with clauses $c_1, \ldots, c_m$, and variables $x_1, \ldots, x_n$. It will be convenient for us to denote TRUE and FALSE by -1 and 1, respectively. For a truth assignment $x = (x_1, \ldots, x_n) \in \{-1, 1\}^n$, the *excess* $\epsilon_F(x)$ for x is the number of clauses satisfied by x minus $\mathrm{asat}(x)$. Thus, the answer to (F, k) is YES if and only if there is an assignment x with $\epsilon_F(x) \geq k$.

MAX-$r(n)$-SAT-AA is related to MAX-$r(n)$-LIN2-AA as follows. Results similar to Lemma 19 have been proved in the previous papers [3, 25].

Lemma 19. *Let (F,k) be an instance of* Max-$r(n)$-SAT-AA *with n variables, m clauses and parameter k, and on clause contains both a literal and its complement. Then, in time $2^{r(n)}m^{O(1)}$, we can produce an instance (S',k') of* Max-$r(n)$-Lin2-AA *reduced by Rule 4.1, with parameter $k'=k2^{r(n)-1}$ such that (F,k) is a* Yes*-instance if and only if (S',k') is a* Yes*-instance. Moreover, F and S' have the same set of variables and for any truth assignment $x=(x_1,\ldots,x_n)\in\{-1,1\}^n$, $\epsilon_{S'}(x)=\varepsilon_F(x)\cdot 2^{r(n)-1}$.*

Proof. Let (F,k) be an instance of Max-$r(n)$-SAT-AA with clauses $c_1,\ldots,c_m$ and variables $x_1,\ldots,x_n$. For a clause c_j, $\mathrm{var}(c_j)$ will denote the set of variables in c_j and r_j the number of literals in c_j. For every $j\in[m]$, let $x=(x_1,\ldots,x_n)\in\{-1,1\}^n$ and

$$h_j(x)=2^{r(n)-r_j}[1-\prod_{x_i\in\mathrm{var}(c_j)}(1+d_{ij}x_i)],$$

where $d_{ij}=1$ if x_i is in c_j and $d_{ij}=-1$ if $\overline{x}_i$ is in c_j.

Let $H(x)=\sum_{j=1}^m h_j(x)$. Let $x=(x_1,\ldots,x_n)\in\{-1,1\}^n$ be a fixed truth assignment. We will prove that

$$H(x)=2^{r(n)}\epsilon_F(x). \tag{6.1}$$

Let $q_j=1$ if c_j is satisfied by x and $q_j=0$, otherwise. Observe that $h_j(x)/(2^{r(n)-r_j})$ equals $1-2^{r_j}$ if $q_j=0$ and 1, otherwise. Thus,

$$\begin{aligned} H(x) &= \textstyle\sum_{j=1}^m[2^{r(n)-r_j}q_j+(2^{r(n)-r_j}-2^{r(n)})(1-q_j)] \\ &= 2^{r(n)}[\textstyle\sum_{j=1}^m q_j-\sum_{j=1}^m(1-2^{-r_j})] \\ &= 2^{r(n)}\epsilon_F(x). \end{aligned}$$

It follows from (6.1) that the answer to (F,k) is Yes if and only if there is a truth assignment x such that

$$H(x)\geq k2^{r(n)}. \tag{6.2}$$

Algebraic simplification of $H(x)$ will lead us to the following (where c_L are the nonzero integers resulting from this simplification[1]):

$$H(x)=\sum_{L\in\mathcal{F}}c_L\prod_{i\in L}x_i, \tag{6.3}$$

where $\mathcal{F}\subseteq\{\emptyset\neq L\subseteq[n]:\ |L|\leq r(n)\}$. The simplification can be done in time $2^{r(n)}m^{O(1)}$.

Observe that by replacing each term $c_L\prod_{i\in L}x_i$ with the equation $\prod_{i\in L}x_i=1$ if $c_L>0$ and $\prod_{i\in L}x_i=-1$ if $c_L<0$, with weight $|c_L|$, the sum $\sum_{L\in\mathcal{F}}c_L\prod_{i\in L}x_i$ can be viewed as twice the excess for x of an instance (S',k') of Max-$r(n)$-Lin2-AA. Let $k'=k2^{r(n)-1}$ be the parameter of (S',k'). Then, by (6.2), (F,k) and (S',k') are equivalent.

[1]Formula (6.3) is the Fourier expansion of $H(x)$ over basis $\prod_{i\in L}, L\subseteq[n]$, and c_L are nonzero Fourier coefficients of $H(x)$; for more information, see [25, 26]

Note that the algebraic simplification of $H(x)$ ensures that (S', k') is reduced by Rule 4.1. This completes the proof. □

It is important to note that the resulting instance (S', k') of MAX-$r(n)$-LIN2-AA is not necessarily reduced under Rule 4.2 and, thus, reduction of (S', k') by Rule 4.2 may result in less than n variables.

From Theorem 17 and Lemma 19 we have the following fixed-parameter-tractability result for MAX-$r(n)$-SAT-AA.

Theorem 18. MAX-$r(n)$SAT-AA *is (i) in XP for* $r(n) \leq \log\log n - \log\log\log n$ *and (ii) fixed-parameter tractable for* $r(n) \leq \log\log n - \log\log\log n - \varphi(n)$*, for any real-valued unbounded strictly increasing computable function* $\varphi(n)$*.*

Proof. We start by proving Part (ii). Let $\varphi(n)$ be a real-valued unbounded strictly increasing computable function of n. Note that if Part (ii) holds for some unbounded strictly increasing computable function $\psi'(n)$ of n, it holds also for any strictly increasing computable function $\psi(n)$ such that $\psi(n) \geq \psi'(n)$ for every $n \geq 1$. Thus, it suffices to give the proof for $\varphi(n) \leq \log\log\log n$, as for any strictly increasing function $\psi(n)$ we can consider the case when $\varphi(n) = \min\{\psi(n), \log\log\log n, \varphi(n-1)+1\}$.

Observe that $\log\log n - \log\log\log n - \varphi(n)$ is a nondecreasing function starting from some value n_0' of n. Let n_0'' be the minimum value of n such that $\varphi(n) > 0$ for all $n \geq n_0''$ and let $n_0 = \max\{n_0', n_0''\}$.

If $n \leq n_0$, the problem is trivially solvable in polynomial time. Thus, we may assume that $n > n_0$ and so $\varphi(n) > 0$ and $\log\log n - \log\log\log n - \varphi(n)$ is nondecreasing. Note that $\varphi(n)$ can be extended to a continuous positive strictly increasing function $\varphi(t)$ of real argument $t \geq 1$. Thus, $\varphi(t)$ has an inverse function $\varphi^{-1}(t)$.

Let $r(n) = \lfloor \log\log n - \log\log\log n - \varphi(n) \rfloor$ and consider an instance (F^*, k) of MAX-$r(n)$-SAT-AA. From this form an equivalent instance $(F, k-\delta)$ of MAXSAT-AA where each clause contains at most $r(n)$ variables, and no clause contains a literal and it's negation. Note that $2^{r(n)} \leq n$. Therefore by Lemma 19, in polynomial time we can reduce (F, k) into an instance (S', k') of MAX-LIN2-AA where each equation contains at most $r(n)$ variables, such that $(F, k-\delta)$ is a YES-instance if and only if (S', k') is a YES-instance with parameter $k' = (k-\delta) \cdot 2^{r(n)-1}$. Consider the MAXLIN2-AA instance (S'', k') with n' variables formed by reducing (S', k') by Rules 4.2 and 4.1. If $n' \leq \log n$, (S'', k') may be solved in polynomial time by trying all $2^{n'} \leq n$ assignments to the variables of S''. Thus, we may assume that $n' > \log n$. We may also assume that $n' > n_0$.

Note that no equation of S'' has more than $r(n)$ variables. If $n' \geq ((k-\delta)2^{r(n)} - 1)r(n) + 1$ then, by Theorem 17 and Lemma 19, $(F, k-\delta)$ is a YES-instance. Hence, it remains to consider the case $\log n < n' \leq ((k - -\delta)2^{r(n)} - 1)r(n) \leq (k2^{r(n)} - 1)r(n)$. Thus,

$$\log n \leq (k2^{r(n)} - 1)r(n) \text{ and so } \log n \leq k(\log\log n) \cdot \log n/(2^{\varphi(n)} \log\log n).$$

This simplifies to $\varphi(n) \leq \log k$ and so $n \leq \varphi^{-1}(\log k)$. Hence, $(F, k-\delta)$ can be solved in time $2^{\varphi^{-1}(\log k)}m^{O(1)}$ by trying all possible assignments to variables of (S'', k').

Now we will prove Part (i). Let $r(n) = \lfloor \log\log n - \log\log\log n \rfloor$ and consider an instance (F^*, k) of MAX-$r(n)$-SAT-AA. As in the proof of Part (ii), we reduce (F^*, k) into $(F, k-\delta)$ into an instance (S', k') of MAX-$r(n)$-LIN2-AA, such that $(F*, k)$ is a YES-instance if and only if (S', k') is a YES-instance with parameter $k' = (k-\delta)\cdot 2^{r(n)-1}$. Consider the MAXLIN2-AA instance (S'', k') with n' variables formed by reducing (S', k') by Rules 4.2 and 4.1. If $n' \leq k\log n$, (S'', k') may be solved in XP time by trying all $2^{n'} \leq n^k$ assignments to the variables of S''. Thus, we may assume that $n' > k\log n$. If $n' \geq ((k-\delta)2^{r(n)} - 1)r(n) + 1$, then by Theorem 17 and Lemma 19, (F, k) is a YES-instance. Thus, it remains to consider the case $k\log n < n' \leq ((k-\delta)2^{r(n)} - 1)r(n) \leq (k2^{r(n)} - 1)r(n)$. We have $k\log n \leq kr(n)2^{r(n)}$ and so $\log\log n \leq r(n) + \log r(n) < \log\log n$ (as $\log r(n) < \log\log\log n$), a contradiction. Thus, this case is impossible and we can solve $(F*, k)$ in XP time. □

We conclude this section with a table summarizing the results:

Constraint	Results	Assumption
$r(n) \leq \log\log n - \log\log\log n - \varphi(n)$	is FPT	
$r(n) \leq \log\log n - \log\log\log n$	is in XP	
$r(n) \geq \log\log n + \varphi(n)$	Not in XP	ETH
$r(n) \geq \lceil \log n \rceil$	para-NP-complete	

Chapter 7

MaxSAT Above the number of variables

7.1 Introduction

Given a CNF formula F, let B_F denote the bipartite graph with partite sets $V(F)$ and F with an edge between $v \in V(F)$ and $c \in F$ if $v \in V(c)$. The *matching number* $\nu(F)$ of F is the size of a maximum matching in B_F. Clearly, $\text{sat}(F) \geq \nu(F)$ and this lower bound for $\text{sat}(F)$ is tight as there are formulas F for which $\text{sat}(F) = \nu(F)$. In this chapter we study the following parameterized problem, where the parameterization is above a tight lower bound.

MaxSat-A$\nu(F)$
Instance: A CNF formula F and a positive integer α.
Parameter: $k = \alpha - \nu(F)$.
Question: Is $\text{sat}(F) \geq \alpha$?

In our main result, we show that MaxSat-A$\nu(F)$ is fixed-parameter tractable by obtaining an algorithm with running time $O((2e)^{2k+O(\log^2 k)}(n+m)^{O(1)})$, where e is the base of the natural logarithm. We also develop a randomized algorithm for MaxSat-A$\nu(F)$ of expected runtime $O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})$.

The *deficiency* $\delta(F)$ of a formula F is $|F|-|V(F)|$; the *maximum deficiency* $\delta^*(F) = \max_{F' \subseteq F} \delta(F')$. A formula F is called *variable-matched* if $\nu(F) = |V(F)|$. Our main result implies fixed-parameter tractability of MaxSat parameterized by $\delta(F)$ for variable-matched formulas F.

To obtain our main result, we introduce some reduction rules and branching steps and reduce the problem to a parameterized version of Hitting Set, namely, $(m-k)$-Hitting Set defined

below. Let H be a hypergraph. A set $S \subseteq V(H)$ is called a *hitting set* if $e \cap S \neq \emptyset$ for all $e \in E(H)$.

$(m-k)$-Hitting Set
Instance: A hypergraph H $(n = \|V(H)\|,\ m = \|E(H)\|)$ and a positive integer k.
Parameter: k.
Question: Does there exist a hitting set $S \subseteq V(H)$ of size $m-k$?

Gutin *et al.* [45] showed that $(m-k)$-Hitting Set is fixed-parameter tractable by obtaining a kernel for the problem. The kernel result immediately implies a $2^{O(k^2)}(m+n)^{O(1)}$-time algorithm for the problem. Here we obtain a faster algorithm for this problem that runs in $O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$ time using the color-coding technique. This happens to be the dominating step for solving the MaxSat-A$\nu(F)$ problem. We also obtain a randomized algorithm for $(m-k)$-Hitting Set of expected runtime $O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})$. To obtain the randomized algorithm, we reduce $(m-k)$-Hitting Set into a special case of the Subgraph Isomorphism problem and use a recent randomized algorithm of Fomin *et al.* [42] for Subgraph Isomorphism.

It was shown in [45] that the $(m-k)$-Hitting Set problem cannot have a kernel whose size is polynomial in k unless NP $\subseteq$ coNP/poly. In this chapter, we give a parameter preserving reduction from this problem to the MaxSat-A$\nu(F)$ problem, thereby showing that MaxSat-A$\nu(F)$ problem has no polynomial-size kernel unless NP $\subseteq$ coNP/poly.

In Section 7.2, we give a sequence of polynomial time preprocessing rules on the given input of MaxSat-A$\nu(F)$ and justify their correctness. In Section 7.3, we give two simple branching rules and reduce the resulting input to a $(m-k)$-Hitting Set problem instance. Section 7.4 gives an improved fixed-parameter algorithm for $(m-k)$-Hitting Set using color coding. There we also obtain a faster randomized algorithm for $(m-k)$-Hitting Set. Section 7.5 summarizes the entire algorithm for the MaxSat-A$\nu(F)$ problem, shows its correctness and analyzes its running time. Section 7.6 proves the hardness of kernelization result.

7.2 Preprocessing Rules

In this section we give preprocessing rules and their correctness.

Let F be the given CNF formula on n variables and m clauses with a maximum matching M on B_F, the variable-clause bipartite graph corresponding to F. Let α be a given integer and recall that our goal is to check whether $\text{sat}(F) \geq \alpha$. For each preprocessing rule below, we let (F', α') be the instance resulting by the application of the rule on (F, α). We say that a rule is *valid* if (F, α) is a Yes instance if and only if (F', α') a Yes instance. We use $n(x)$ to denote the number of occurrences of literal x.

Reduction Rule 7.1. *Let x be a variable such that $n(x) = 0$ (respectively $n(\bar{x}) = 0$). Set*

$x = \text{FALSE}$ *($x = \text{TRUE}$) and remove all the clauses that contain $\bar{x}$ (x). Reduce α by $n(\bar{x})$ (respectively $n(x)$).*

The proof of the following lemma is immediate.

Lemma 20. *If $n(x) = 0$ (respectively $n(\bar{x}) = 0$) then $\text{sat}(F) = \text{sat}(F') + n(\bar{x})$ (respectively $\text{sat}(F) = \text{sat}(F') + n(x)$), and so Rule 7.1 is valid.*

Reduction Rule 7.2. *Let $n(x) = n(\bar{x}) = 1$ and let c' and c'' be the two clauses containing x and $\bar{x}$, respectively. Let $c^* = (c' - x) \cup (c'' - \bar{x})$ and let F' be obtained from F be deleting c' and c'' and adding the clause c^*. Reduce α by 1.*

Lemma 21. *For F and F' in Reduction Rule 7.2, $\text{sat}(F) = \text{sat}(F') + 1$, and so Rule 7.2 is valid.*

Proof. Consider any assignment for F. If it satisfies both c' and c'', then the same assignment will satisfy c^*. So when restricted to variables of F', it will satisfy at least $sat(F) - 1$ clauses of F'. Thus $sat(F') \geq sat(F) - 1$ which is equivalent to $sat(F) \leq sat(F') + 1$. Similarly if an assignment γ to F' satisfies c^* then at least one of c', c'' is satisfied by γ. Therefore by setting x true if γ satisfies c'' and false otherwise, we can extend γ to an assignment on F that satisfies both of c', c''. On the other hand, if c^* is not satisfied by γ then neither c' nor c'' is satisfied by γ, and any extension of γ will satisfy exactly one of c', c''. Therefore in either case $sat(F) \geq sat(F') + 1$. We conclude that $\text{sat}(F) = \text{sat}(F') + 1$, as required. □

Our next reduction rule is based on the following lemma proved in Fleischner *et al.* [39, Lemma 10], Kullmann [78, Lemma 7.7] and Szeider [103, Lemma 9].

Lemma 22. *Let F be a CNF formula. Given a maximum matching in B_F, in time $O(|F|)$ we can find an autarky $L : U \to \{\text{TRUE}, \text{FALSE}\}$ such that $F \setminus F_U$ is 1-expanding.*

Reduction Rule 7.3. *Find an autarky $L : U \to \{\text{TRUE}, \text{FALSE}\}$ such that $F \setminus F_U$ is 1-expanding. Set $F' = F \setminus F_U$ and reduce α by $|F_U|$.*

The next lemma follows from Lemma 5.

Lemma 23. *For F and F' in Reduction Rule 7.3, $\text{sat}(F) = \text{sat}(F') + |F_U|$ and so Rule 7.3 is valid.*

After exhaustive application of Rule 7.3, we may assume that the resulting formula is 1-expanding. For the next reduction rule, we need the following results.

Theorem 19 (Szeider [103])**.** *Given a variable-matched formula F, with $|F| = |V(F)| + 1$, we can decide whether F is satisfiable in time $O(|V(F)|^3)$.*

Consider a bipartite graph $G = (A, B; E)$. Recall that a formula F is q-expanding if and only if B_F is q-expanding. From a bipartite graph $G = (A, B; E)$, $x \in A$ and $q \geq 1$, we obtain a bipartite graph G_{qx}, by adding new vertices $x_1, \ldots, x_q$ to A and adding edges such that new vertices have exactly the same neighborhood as x, that is, $G_{qx} = (A \cup \{x_1, \ldots, x_q\}, B; E \cup \{(x_i, y) : (x, y) \in E\})$. The following result is well known.

Lemma 24. [84, Theorem 1.3.6] *Let $G = (A, B; E)$ be a 0-expanding bipartite graph. Then G is q-expanding if and only if G_{qx} is 0-expanding for all $x \in A$.*

Lemma 25. *Let $G = (A, B; E)$ be a 1-expanding bipartite graph. In polynomial time, we can check whether G is 2-expanding, and if it is not, find a set $S \subseteq A$ such that $|N_G(S)| = |S| + 1$.*

Proof. Let $x \in A$. By Hall's Matching Theorem, G_{2x} is 0-expanding if and only if $\nu(G_{2x}) = |A|+2$. Since we can check the last condition in polynomial time, by Lemma 24 we can decide whether G is 2-expanding in polynomial time. So, assume that G is not 2-expanding and we know this because G_{2y} is not 0-expanding for some $y \in A$. By Lemma 3(4) in [103], in polynomial time, we can find a set $T \subseteq A \cup \{y_1, y_2\}$ such that $|N_{G_{2y}}(T)| < |T|$. Since G is 1-expanding, $y_1, y_2 \in T$ and $|N_{G_{2y}}(T)| = |T| - 1$. Hence, $|S| + 1 = |N_G(S)|$, where $S = T \setminus \{y_1, y_2\}$. □

For a formula F and a set $S \subseteq V(F)$, $F[S]$ denotes the formula obtained from F_S by deleting all variables not in S.

Reduction Rule 7.4. *Let F be a 1-expanding formula and let $B = B_F$. Using Lemma 25, check whether F is 2-expanding. If it is then do not change F, otherwise find a set $S \subseteq V(F)$ with $|N_B(S)| = |S| + 1$. Use Theorem 19 to decide whether $F[S]$ is satisfiable, and proceed as follows.*

$F[S]$ **is satisfiable:** *Obtain a new formula F' by removing all clauses in $N_B(S)$ from F. Reduce α by $|N_B(S)|$.*

$F[S]$ **is not satisfiable:** *Delete all variables in S from $\bigcup_{c'' \in N_B(S)} c''$ to form the clause c'. That is, a literal l belongs to c' if and only if it belongs to some clause in $N_B(S)$ and the variable corresponding to l is not in S. Obtain a new formula F' by removing all clauses in $N_B(S)$ from F and adding c'. Reduce α by $|S|$.*

Lemma 26. *For F, F' and S introduced in Rule 7.4, if $F[S]$ is satisfiable* $\mathrm{sat}(F) = \mathrm{sat}(F') + |N_B(S)|$, *otherwise* $\mathrm{sat}(F) = \mathrm{sat}(F') + |S|$ *and thus Rule 7.4 is valid.*

Proof. We consider two cases.
Case 1: $F[S]$ **is satisfiable.** Observe that there is an autarky on S and thus by Lemma 5, $\mathrm{sat}(F) = \mathrm{sat}(F') + |N_B(S)|$.
Case 2: $F[S]$ **is not satisfiable.** Let $F'' = F' \setminus c'$. As any optimal truth assignment to F will satisfy at least $\mathrm{sat}(F) - |N_B(S)|$ clauses of F'', it follows that $\mathrm{sat}(F) \leq \mathrm{sat}(F'') + |N_B(S)| \leq \mathrm{sat}(F') + |N_B(S)|$.

Let M be a matching that saturates S in $B[S \cup N_B(S)]$ (that exists as $B[S \cup N_B(S)]$ is 1-expanding), and let y denote the clause in $N_B(S)$ that is not matched to a variable in S by M. Let S' be the set of variables, and Z the set of clauses, that can be reached from y with an M-alternating path in $B[S \cup N_B(S)]$. We argue now that $Z = N_B(S)$. Since Z is made up of clauses that are reachable in $B[S \cup N_B(S)]$ by an M-alternating path from the single unmatched clause y, $|Z| = |S'| + 1$. It follows that $|N_B(S) \setminus Z| = |S \setminus S'|$, and M matches every clause in $N_B(S) \setminus Z$ with

a variable in $S \setminus S'$. Furthermore, $N_B(S \setminus S') \cap Z = \emptyset$ as otherwise the matching partners of some elements of $S \setminus S'$ would have been reachable by an M-alternating path from y, contradicting the definition of $N_B(S)$ and S'. Thus $S \setminus S'$ has an autarky such that $F \setminus F_{S \setminus S'}$ is 1-expanding which would have been detected by Rule 7.3, hence $S \setminus S' = \emptyset$ and so $S = S'$. That is, all clauses in $N_B(S)$ are reachable from the unmatched clause y by an M-alternating path. We have now shown that $Z = N_B(S)$, as desired.

Suppose that there exists an assignment γ to F', that satisfies $\text{sat}(F')$ clauses of F' that also satisfies c'. Then there exists a clause $c'' \in N_B(S)$ that is satisfied by γ. As c'' is reachable from y by an M-alternating path, we can modify M to include y and exclude c'', by taking the symmetric difference of the matching and the M-alternating path from y to c''. This will give a matching saturating S and $N_B(S) \setminus c''$, and we use this matching to extend the assignment γ to one which satisfies all of $N_B(S) \setminus c''$. We therefore have satisfied all the clauses of $N_B(S)$. Therefore since c' is satisfied in F' but does not appear in F, we have satisfied extra $|N_B(S)| - 1 = |S|$ clauses. Suppose on the other hand that every assignment γ for F' that satisfies $\text{sat}(F')$ clauses does not satisfy c'. We can use the matching on $B[S \cup N_B(S)]$ to satisfy $|N_B(S)| - 1$ clauses in $N_B(S)$, which would give us an additional $|S|$ clauses in $N_B(S)$. Thus $\text{sat}(F) \geq \text{sat}(F') + |S|$.

As $|N_B(S)| = |S| + 1$, it suffices to show that $\text{sat}(F) < \text{sat}(F') + |N_B(S)|$. Suppose that there exists an assignment γ to F that satisfies $\text{sat}(F') + |N_B(S)|$ clauses, then it must satisfy all the clauses of $N_B(S)$ and $\text{sat}(F')$ clauses of F''. As $F[S]$ is not satisfiable, variables in S alone can not satisfy all of $N_B(S)$. Hence there exists a clause $c'' \in N_B(S)$ such that there is a variable $v \in V(c'') \setminus S$ that satisfies c''. But then $v \in V(c')$ and hence c' would be satisfiable by γ, a contradiction as γ satisfies $\text{sat}(F')$ clauses of F''. □

7.3 Branching Rules and Reduction to $(m-k)$-Hitting Set

Our algorithm first applies Reduction Rules 7.1, 7.2, 7.3 and 7.4 exhaustively on (F, α). Then it applies two branching rules we describe below, in the following order.

Branching on a variable x means that the algorithm constructs two instances of the problem, one by substituting $x = \text{TRUE}$ and simplifying the instance and the other by substituting $x = \text{FALSE}$ and simplifying the instance. Branching on x or y being false means that the algorithm constructs two instances of the problem, one by substituting $x = \text{FALSE}$ and simplifying the instance and the other by substituting $y = \text{FALSE}$ and simplifying the instance. Simplifying an instance is done as follows. For any clause c, if c contains a literal z with $z = \text{TRUE}$, remove c and reduce α by 1. If c contains a literal z with $z = \text{FALSE}$ and c contains other literals, remove z from c. If c consists of the single literal $z = \text{FALSE}$, remove c.

A branching rule is correct if the instance on which it is applied is a Yes-instance if and only if the simplified instance of (at least) one of the branches is a Yes-instance.

Branching Rule 7.1. *If $n(x) \geq 2$ and $n(\bar{x}) \geq 2$ then we branch on x.*

Before attempting to apply Branching Rule 7.2, we apply the following rearranging step: For all variables x such that $n(\bar{x}) = 1$, swap literals x and $\bar{x}$ in all clauses. Clearly, this will not change $\text{sat}(F)$. Observe that now for every variable $n(x) = 1$ and $n(\bar{x}) \geq 2$.

Branching Rule 7.2. *If there is a clause c such that positive literals $x, y \in c$ then we branch on x being false or y being false.*

Branching Rule 7.1 is exhaustive and thus its correctness also follows. When we reach Branching Rule 7.2 for every variable $n(x) = 1$ and $n(\bar{x}) \geq 2$. As $n(x) = 1$ and $n(y) = 1$ we note that c is the only clause containing these literals. Therefore there exists an optimal solution with x or y being false (if they are both true just change one of them to false). Thus, we have the following:

Lemma 27. *Branching Rules 7.1 and 7.2 are correct.*

Let (F, α) be the given instance on which Reduction Rules 7.1, 7.2, 7.3 and 7.4, and Branching Rules 7.1 and 7.2 do not apply. Observe that for such an instance F the following holds:

1. For every variable x, $n(x) = 1$ and $n(\bar{x}) \geq 2$.
2. Every clause contains at most one positive literal.

We call a formula F satisfying the above properties *special*. In what follows we describe an algorithm for our problem on special instances. Let $c(x)$ denote the *unique* clause containing positive literal x. We can obtain a matching saturating $V(F)$ in B_F by taking the edge connecting the variable x and the clause $c(x)$. We denote the resulting matching by M_u.

We first describe a transformation that will be helpful in reducing our problem to $(m - k)$-Hitting Set. Given a formula F we obtain a new formula F' by changing the clauses of F as follows. If there exists some $c(x)$ such that $|c(x)| \geq 2$, do the following. Let $c' = c(x) - x$ (that is, c' contain the same literals as $c(x)$ except for x) and add c' to all clauses containing the literal $\bar{x}$. Furthermore remove c' from $c(x)$ (which results in $c(x) = (x)$ and therefore $|c(x)| = 1$).

Next we prove the validity of the above transformation.

Lemma 28. *Let F' be the formula obtained by applying the transformation described above on F. Then $\text{sat}(F') = \text{sat}(F)$ and $\nu(B_F) = \nu(B_{F'})$.*

Proof. We note that the matching M_u remains a matching in $B_{F'}$ and thus $\nu(B_F) = \nu(B_{F'})$. Let γ be any truth assignment to the variables in F (and F') and note that if c' is false under γ then F and F' satisfy exactly the same clauses under γ (as we add and subtract something false to the clauses). So assume that c' is true under γ.

If γ maximizes the number of satisfied clauses in F then clearly we may assume that x is false (as $c(x)$ is true due to c'). Now let γ' be equal to γ except the value of x has been flipped to true. Note that exactly the same clauses are satisfied in F and F' by γ and γ', respectively. Analogously, if an assignment maximizes the number of satisfied clauses in F' we may assume that x is true and by changing it to false we satisfy equally many clauses in F. Hence, $\text{sat}(F') = \text{sat}(F)$. □

Given a special instance (F, α) we apply the above transformation repeatedly until no longer possible and obtain an instance (F', α) such that $\text{sat}(F') = \text{sat}(F)$, $\nu(B_F) = \nu(B_{F'})$ and $|c(x)| = 1$ for all $x \in V(F')$. We call such an instance (F', α) *transformed special*. Observe that, it takes polynomial time, to obtain the transformed special instance from a given special instance.

For simplicity of presentation we denote the transformed special instance by (F, α). Let C^* denote all clauses that are not matched by M_u (and therefore only contain negated literals). We associate a hypergraph H^* with the transformed special instance. Let H^* be the hypergraph with vertex set $V(F)$ and edge set $E^* = \{V(c) \mid c \in C^*\}$.

We now show the following equivalence between MaxSat-A$\nu(F)$ on transformed special instances and $(m-k)$-Hitting Set.

Lemma 29. *Let (F, α) be the transformed special instance and H^* be the hypergraph associated with it. Then $\text{sat}(F) \geq \alpha$ if and only if there is a hitting set in H^* of size at most $|E(H^*)| - k$, where $k = \alpha - \nu(F)$.*

Proof. We start with a simple observation about an assignment satisfying the maximum number of clauses of F. There exists an optimal truth assignment to F, such that all clauses in C^* are true. Assume that this is not the case and let γ be an optimal truth assignment satisfying as many clauses from C^* as possible and assume that $c \in C^*$ is not satisfied. Let $\bar{x} \in c$ be an arbitrary literal and note that $\gamma(x) = \text{TRUE}$. However, changing x to false does not decrease the number of satisfied clauses in F and increases the number of satisfied clauses in C^*.

Now we show that $\text{sat}(F) \geq \alpha$ if and only if there is a hitting set in H^* of size at most $|E(H^*)| - k$. Assume that γ is an optimal truth assignment to F, such that all clauses in C^* are true. Let $U \subseteq V(F)$ be all variables that are false in γ and note that U is a hitting set in H^*. Analogously if U' is a hitting set in H^* then by letting all variables in U' be false and all other variables in $V(F)$ be true we get a truth assignment that satisfies $|F| - |U'|$ clauses in F. Therefore if $\tau(H^*)$ is the size of a minimum hitting set in H^* we have $\text{sat}(F) = |F| - \tau(H^*)$. Hence, $\text{sat}(F) = |F| - \tau(H^*) = |V(F)| + |C^*| - \tau(H^*)$ and thus $\text{sat}(F) \geq \alpha$ if and only if $|C^*| - \tau(H^*) \geq k$, which is equivalent to $\tau(H^*) \leq |E(H^*)| - k$. □

Therefore our problem is fixed-parameter tractable on transformed special instances, by the next theorem that follows from the kernelization result in [45].

Theorem 20. *There exists an algorithm for $(m-k)$-Hitting Set running in time $2^{O(k^2)} + O((n+m)^{O(1)})$.*

In the next section we give faster algorithms for MaxSat-A$\nu(F)$ on transformed special instances by giving faster algorithms for $(m-k)$-Hitting Set.

7.4 Algorithms for $(m-k)$-Hitting Set

To obtain faster algorithms for $(m-k)$-Hitting Set, we utilize the following concept of *k-mini-hitting set* introduced in [45].

Definition 1. *Let $H = (V, \mathcal{F})$ be a hypergraph and k be a nonnegative integer. A* k-mini-hitting set *is a set $S_{\text{MINI}} \subseteq V$ such that $|S_{\text{MINI}}| \leq k$ and $|\mathcal{F}[S_{\text{MINI}}]| \geq |S_{\text{MINI}}| + k$.*

Lemma 30 ([45])**.** *A hypergraph H has a hitting set of size at most $m - k$ if and only if it has a k-mini-hitting set. Moreover, given a k-mini-hitting set S_{MINI}, we can construct a hitting set S with $|S| \leq m - k$ such that $S_{\text{MINI}} \subseteq S$ in polynomial time.*

7.4.1 Deterministic Algorithm

Next we give an algorithm that finds a k-mini-hitting set S_{MINI} if it exists, in time $c^k(m+n)^{O(1)}$, where c is a constant. We first describe a randomized algorithm based on color-coding [6] and then derandomize it using hash functions. Let $\chi : E(H) \to [q]$ be a function. For a subset $S \subseteq V(H)$, $\chi(S)$ denotes the maximum subset $X \subseteq [q]$ such that for all $i \in X$ there exists an edge $e \in E(H)$ with $\chi(e) = i$ and $e \cap S \neq \emptyset$. A subset $S \subseteq V(H)$ is called a *colorful hitting set* if $\chi(S) = [q]$. We now give a procedure that given a coloring function χ finds a minimum colorful hitting set, if it exists. This algorithm will be useful in obtaining a k-mini-hitting set S_{MINI}.

Lemma 31. *Given a hypergraph H and a coloring function $\chi : E(H) \to [q]$, we can find a minimum colorful hitting set if there exists one in time $O(2^q q(m+n))$.*

Proof. We first check whether for every $i \in [q]$, $\chi^{-1}(i) \neq \emptyset$. If for any i we have that $\chi^{-1}(i) = \emptyset$, then we return that there is no colorful hitting set. So we may assume that for all $i \in [q]$, $\chi^{-1}(i) \neq \emptyset$. We will give an algorithm using dynamic programming over subsets of $[q]$. Let γ be an array of size 2^q indexed by the subsets of $[q]$. For a subset $X \subseteq [q]$, let $\gamma[X]$ denote the size of a smallest set $W \subseteq V(H)$ such that $X \subseteq \chi(W)$. We obtain a recurrence for $\gamma[X]$ as follows:

$$\gamma[X] = \begin{cases} \min_{(v \in V(H), \chi(\{v\}) \cap X \neq \emptyset)} \{1 + \gamma[X \setminus \chi(\{v\})]\} & \text{if } |X| \geq 1, \\ 0 & \text{if } X = \emptyset. \end{cases}$$

The correctness of the above recurrence is clear. The algorithm computes $\gamma[[q]]$ by filling the γ in the order of increasing set sizes. Clearly, each cell can be filled in time $O(q(n+m))$ and thus the whole array can be filled in time $O(2^q q(n+m))$. The size of a minimum colorful hitting set is given by $\gamma[[q]]$. We can obtain a minimum colorful hitting set by the routine back-tracking. □

Now we describe a randomized procedure to obtain a k-mini-hitting set S_{MINI} in a hypergraph H, if there exists one. We do the following for each possible value p of $|S_{\text{MINI}}|$ (that is, for $1 \leq p \leq k$). Color $E(H)$ uniformly at random with colors from $[p+k]$; we denote this random coloring by χ. Assume that there is a k-mini-hitting set S_{MINI} of size p and some $p+k$ edges $e_1, \ldots, e_{p+k}$ such that for all $i \in [p+k]$, $e_i \cap S_{\text{MINI}} \neq \emptyset$. The probability that for all $1 \leq i < j \leq p+k$ we have that $\chi(e_i) \neq \chi(e_j)$ is $\frac{(p+k)!}{(p+k)^{p+k}} \geq e^{-(p+k)} \geq e^{-2k}$. Now, using Lemma 31 we can test in time $O(2^{p+k}(p+k)(m+n))$ whether there is a colorful hitting set of size at most p. Thus with probability at least e^{-2k} we can find a S_{MINI}, if there exits one. To boost the probability we repeat the procedure e^{2k} times and thus in time $O((2e)^{2k}2k(m+n)^{O(1)})$ we find a S_{MINI}, if there exists

one, with probability at least $1-(1-\frac{1}{e^{2k}})^{e^{2k}} \geq \frac{1}{2}$. If we obtained S_{MINI} then using Lemma 30 we can construct a hitting set of H of size at most $m-k$.

To derandomize the procedure, we need to replace the first step of the procedure where we color the edges of $E(H)$ uniformly at random from the set $[p+k]$ to a deterministic one. This is done by making use of an $(m, p+k, p+k)$-*perfect hash family*. An $(m, p+k, p+k)$-perfect hash family, $\mathcal{H}$, is a set of functions from $[m]$ to $[p+k]$ such that for every subset $S \subseteq [m]$ of size $p+k$ there exists a function $f \in \mathcal{H}$ such that f is injective on S. That is, for all $i, j \in S$, $f(i) \neq f(j)$. There exists a construction of an $(m, p+k, p+k)$-perfect hash family of size $O(e^{p+k} \cdot k^{O(\log k)} \cdot \log m)$ and one can produce this family in time linear in the output size [102]. Using an $(m, p+k, p+k)$-perfect hash family $\mathcal{H}$ of size at most $O(e^{2k} \cdot k^{O(\log k)} \cdot \log m)$ rather than a random coloring we get the desired deterministic algorithm. To see this, it is enough to observe that if there is a subset $S_{\text{mini}} \subseteq V(H)$ such that $|\mathcal{F}[S_{\text{MINI}}]| \geq |S_{\text{MINI}}| + k$ then there exists a coloring $f \in \mathcal{H}$ such that the $p+k$ edges $e_1, \ldots, e_{p+k}$ that intersect S_{mini} are distinctly colored. So if we generate all colorings from $\mathcal{H}$ we will encounter the desired f. Hence for the given f, when we apply Lemma 31 we get the desired result. This concludes the description. The total time of the derandomized algorithm is $O(k2^{2k}(m+n)e^{2k} \cdot k^{O(\log k)} \cdot \log m) = O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$.

Theorem 21. *There exists an algorithm solving* $(m-k)$-Hitting Set *in time* $O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$.

By Theorem 21 and the transformation discussed in Section 7.3 we have the following theorem.

Theorem 22. *There exists an algorithm solving a transformed special instance of* MaxSat-A$\nu(F)$ *in time* $O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$.

7.4.2 Randomized Algorithm

In this subsection we give a randomized algorithm for $(m-k)$-Hitting Set running in time $O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})$. However, unlike the algorithm presented in the previous subsection we do not know how to derandomize this algorithm. Essentially, we give a randomized algorithm to find a k-mini-hitting set S_{MINI} in the hypergraph H, if it exists.

Towards this we introduce notions of a star-forest and a bush. We call $K_{1,\ell}$ a *star of size* ℓ; a vertex of degree ℓ in $K_{1,\ell}$ is a *central vertex* (thus, both vertices in $K_{1,1}$ are central). A *star-forest* is a forest consisting of stars. A star-forest F is said to have *dimension* $(a_1, a_2, \ldots, a_p)$ if F has p stars with sizes a_1, a_2, ..., a_p respectively. Given a star-forest F of dimension $(a_1, a_2, \ldots, a_p)$, we construct a graph, which we call a *bush of dimension* $(a_1, a_2, \ldots, a_p)$, by adding a triangle (x, y, z) and making y adjacent to a central vertex of in every star of F.

For a hypergraph $H = (V, \mathcal{F})$, the *incidence bipartite graph* B_H of H has partite sets V and $\mathcal{F}$, and there is an edge between $v \in V$ and $e \in \mathcal{F}$ in H if $v \in e$. Given B_H, we construct B_H^* by adding a triangle (x, y, z) and making y adjacent to every vertex in the V. The following lemma relates k-mini-hitting sets to bushes.

Lemma 32. *A hypergraph $H = (V, \mathcal{F})$ has a k-mini-hitting set S_{MINI} if and only if there exists a tuple $(a_1, \ldots, a_p)$ such that*

(a) *$p \leq k$, $a_i \geq 1$ for all $i \in [p]$, and $\sum_{i=1}^{p} a_i = p + k$; and*

(b) *there exists a subgraph of B_H^* isomorphic to a bush of dimension $(a_1, \ldots, a_p)$.*

Proof. We first prove that the existence of a k-mini-hitting set in H implies the existence of a bush in B_H^* of dimension satisfying (a) and (b). Let $S_{\text{MINI}} = \{w_1, \ldots, w_q\}$ be a k-mini-hitting set and let $S_i = \{w_1, \ldots, w_i\}$. We know that $q \leq k$ and $|\mathcal{F}[S_{\text{MINI}}]| \geq |S_{\text{MINI}}| + k$. We define $\mathcal{E}_i := \mathcal{F}[S_i] \setminus \mathcal{F}[S_{i-1}]$ for every $i \geq 2$, and $\mathcal{E}_1 := \mathcal{F}[S_1]$. Let $\mathcal{E}_{s_1}, \ldots, \mathcal{E}_{s_r}$ be the subsequence of the sequence $\mathcal{E}_1, \ldots, \mathcal{E}_q$ consisting only of non-empty sets $\mathcal{E}_i$, and let $b_j = |\mathcal{E}_{s_j}|$ for each $j \in [r]$. Let p be the least integer from $[r]$ such that $\sum_{i=1}^{p} b_i \geq k + p$.

Observe that for every $j \in [p]$, the vertex w_{s_j} belongs to each hyperedge of $\mathcal{E}_{s_j}$. Thus, the bipartite graph B_H contains a star-forest F of dimension $(b_1, \ldots, b_p)$, such that $p \leq k$, $b_j \geq 1$ for all $j \in [p]$, and $c := \sum_{j=1}^{p} b_j \geq p + k$. Moreover, each star in F has a central vertex in V. By the minimality of p, we have $\sum_{j=1}^{p-1} b_j < p - 1 + k$ and so $b_p \geq c + 1 - (p + k)$. Thus, the integers a_j defined as follows are positive: $a_j := b_j$ for every $j \in [p-1]$ and $a_p := b_p - c + (p + k)$. Hence, B_H contains a star-forest F' of dimension $(a_1, \ldots, a_p)$, such that each star in F' has a central vertex in V.

Thus, all central vertices are in V, $p \leq k$, $a_i \geq 1$ for all $i \in [p]$, and $\sum_{i=1}^{p} a_i = p + k$, which implies that B_H^* contains, as a subgraph, a bush with dimension $(a_1, \ldots, a_p)$ satisfying the conditions above.

The construction above relating a k-mini-hitting set of H with the required bush of B_H^* can be easily reversed in the following sense: the existence of a bush of dimension satisfying (a) and (b) in B_H^* implies the existence of a k-mini-hitting set in H. Here the triangle ensures that the central vertices are in V. This completes the proof. □

Next we describe a fast randomized algorithm for deciding the existence of a k-mini-hitting set using the characterization obtained in Lemma 32. Towards this we will use a fast randomized algorithm for the Subgraph Isomorphism problem. In the Subgraph Isomorphism problem we are given two graphs F and G on k and n vertices, respectively, as an input, and the question is whether there exists a subgraph of G isomorphic to F. Recall that $\mathbf{tw}(G)$ denotes the treewidth of a graph G. We will use the following result.

Theorem 23 (Fomin *et al.* [42])**.** *Let F and G be two graphs on q and n vertices respectively and $\mathbf{tw}(F) \leq t$. Then, there is a randomized algorithm for the* Subgraph Isomorphism *problem that runs in expected time $O(2^q (nt)^{t+O(1)})$.*

Let $\mathcal{P}_\ell(s)$ be the set of all *unordered partitions* of an integer s into ℓ parts. Nijenhuis and Wilf [93] designed a polynomial delay generation algorithm for partitions of $\mathcal{P}_\ell(s)$. Let $p(s)$ be the partition function, i.e., the overall number of partitions of s. The asymptotic behavior of $p(s)$ was

first evaluated by Hardy and Ramanujan in the paper in which they develop the famous "circle method."

Theorem 24 (Hardy and Ramanujan [60]). *We have* $p(s) \sim e^{L\sqrt{\frac{2s}{3}}}/(4s\sqrt{3})$, *as* $s \to \infty$.

This theorem and the algorithm of Nijenhuis and Wilf [93] imply the following:

Proposition 2. *There is an algorithm of runtime* $2^{O(\sqrt{s})}$ *for generating all partitions in* $\mathcal{P}_\ell(s)$.

Now we are ready to describe and analyze a fast randomized algorithm for deciding the existence of a k-mini-hitting set in a hypergraph H. By Lemma 32, it suffices to design and analyze a fast randomized algorithm for deciding the existence of a bush in B_H^* of dimension $(a_1, \ldots, a_p)$ satisfying conditions (a) and (b) of Lemma 32. Our algorithm starts by building B_H^*. Then it considers all possible values of p one by one ($p \in [k]$) and generates all partitions in $\mathcal{P}_p(p+k)$ using the algorithm of Proposition 2. For each such partition $(a_1, \ldots, a_p)$ that satisfies conditions (a) and (b) of Lemma 32, the algorithm of Fomin *et al.*[42] mentioned in Theorem 23 decides whether B_H^* contains a bush of dimension $(a_1, \ldots, a_p)$. If such a bush exists, we output Yes and we output No, otherwise.

To evaluate the runtime of our algorithm, observe that the treewidth of any bush is 2 and any bush in Lemma 32 has at most $3k+3$ vertices. This observation, the algorithm above, Theorem 23 and Proposition 2 imply the following:

Theorem 25. *There exists a randomized algorithm solving* $(m-k)$-Hitting Set *in expected time* $O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})$.

This theorem, in turn, implies the following:

Theorem 26. *There exists a randomized algorithm solving a transformed special instance of* MaxSat-A$\nu(F)$ *in expected time* $O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})$.

7.5 Complete Algorithm, Correctness and Analysis

The complete algorithm for an instance (F, α) of MaxSat-A$\nu(F)$ is as follows.

Find a maximum matching M on B_F and let $k = \alpha - |M|$. If $k \leq 0$, return Yes. Otherwise, apply Reduction Rules 7.1 to 7.4, whichever is applicable, in that order and then run the algorithm on the reduced instance and return the answer. If none of the Reduction Rules apply, then apply Branching Rule 7.1 if possible, to get two instances (F', α') and (F'', α''). Run the algorithm on both instances; if one of them returns Yes, return Yes, otherwise return No. If Branching Rule 7.1 does not apply then we rearrange the formula and attempt to apply Branching Rule 7.2 in the same way. Finally if $k > 0$ and none of the reduction or branching rules apply, then we have for all variables x, $n(x) = 1$ and every clause contains at most one positive literal, i.e. (F, α) is a special instance. Then solve the problem by first obtaining the transformed special

instance, then the corresponding instance H^* of $(m-k)$-HITTING SET and solving H^* in time $O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$ as described in Sections 7.3 and 7.4.

Correctness of all the preprocessing rules and the branching rules follows from Lemmata 20, 21, 23, 26 and 27.

Analysis of the algorithm. Let (F, α) be the input instance. Let $\mu(F)=\mu=\alpha-\nu(F)$ be the measure. We will first show that our preprocessing rules do not increase this measure. Following this, we will prove a lower bound on the decrease in the measure occurring as a result of the branching, thus allowing us to bound the running time of the algorithm in terms of the measure μ. For each case, we let (F', α') be the instance resulting by the application of the rule or branch. Also let M' be a maximum matching of $B_{F'}$.

Reduction Rule 7.1: We consider the case when $n(x) = 0$; the other case when $n(\bar{x}) = 0$ is analogous. We know that $\alpha' = \alpha - n(\bar{x})$ and $\nu(F') \geq \nu(F) - n(\bar{x})$ as removing $n(\bar{x})$ clauses can only decrease the matching size by $n(\bar{x})$. This implies that $\mu(F)-\mu(F') = \alpha-\nu(F)-\alpha'+\nu(F') = (\alpha - \alpha') + (\nu(F') - \nu(F)) \geq n(\bar{x}) - n(\bar{x})$. Thus, $\mu(F') \leq \mu(F)$.

Reduction Rule 7.2: We know that $\alpha' = \alpha - 1$. We show that $\nu(F') \geq \nu(F) - 1$. In this case we remove the clauses c' and c'' and add $c^* = (c' - x) \cup (c'' - \bar{x})$. We can obtain a matching of size $\nu(F) - 1$ in $B_{F'}$ as follows. If at most one of the c' and c'' is the end-point of some matching edge in M then removing that edge gives a matching of size $\nu(F) - 1$ for $B_{F'}$. So let us assume that some edges (a, c') and (b, c'') are in M. Clearly, either $a \neq x$ or $b \neq x$. Assume $a \neq x$. Then $M \setminus \{(a, c'), (b, c'')\} \cup \{(a, c^*)\}$ is a matching of size $\nu(F) - 1$ in $B_{F'}$. Thus, we conclude that $\mu(F') \leq \mu(F)$.

Reduction Rule 7.3: The proof is the same as in the case of Reduction Rule 7.1.

Reduction Rule 7.4: The proof that $\mu(F') \leq \mu(F)$ in the case when $F[S]$ is satisfiable is the same as in the case of Reduction Rule 7.1 and in the case when $F[S]$ is not satisfiable is the same as in the case of Reduction Rule 7.2.

Branching Rule 7.1: Consider the case when we set $x =$ TRUE. In this case, $\alpha' = \alpha - n(x)$. Also, since no reduction rules are applicable we have that F is 2-expanding. Hence, $\nu(F) = |V(F)|$. We will show that in (F', α') the matching size will remain at least $\nu(F) - n(x) + 1$ $(= |V(F)| - n(x) + 1 = |V(F')| - n(x) + 2.)$ This will imply that $\mu(F') \leq \mu(F) - 1$. By Lemma 4 and the fact that $n(x) - 2 \geq 0$, it suffices to show that in $B' = B_{F'}$, every subset $S \subseteq V(F')$, $|N_{B'}(S)| \geq |S| - (n(x) - 2)$. The only clauses that have been removed by the simplification process after setting $x =$ TRUE are those where x appears positively and the singleton clauses $(\bar{x})$. Hence, the only edges of $G[S \cup N_B[S]]$ that are in $N_B(S) \setminus N_{B'}(S)$ are those corresponding to clauses that contain x as a pure literal and some variable in S. Thus, $|N_{B'}(S)| \geq |S|+2-n(x) = |S|-(n(x)-2)$ (as F is 2-expanding).

The case when we set $x =$ FALSE is similar to the case when we set $x =$ TRUE. Here, also we can show that $\mu(F') \leq \mu(F) - 1$. Thus, we get two instances, with each instance (F', α') having $\mu(F') \leq \mu(F) - 1$.

Branching Rule 7.2: The analysis here is the same as for Branching Rule 7.1 and again we get two instances with $\mu(F') \leq \mu(F) - 1$.

We therefore have a depth-bounded search tree of size of depth at most $\mu = \alpha - \nu(F) = k$, in which any branching splits an instance into two instances. Thus, the search tree has at most 2^k instances. As each reduction and branching rule takes polynomial time, every rule decreases the number of variables, the number of clauses, or the value of μ, and an instance to which none of the rules apply can be solved in time $O((2e)^{2\mu}\mu^{O(\log \mu)}(m+n)^{O(1)})$ (by Theorem 22), we have by induction that any instance can be solved in time

$$O(2 \cdot (2e)^{2(\mu-1)}(\mu-1)^{O(\log(\mu-1))}(m+n)^{O(1)}) = O((2e)^{2\mu}\mu^{O(\log \mu)}(m+n)^{O(1)}).$$

Thus the total running time of the algorithm is at most $O((2e)^{2k+O(\log^2 k)}(n+m)^{O(1)})$. Applying Theorem 26 instead of Theorem 22, we conclude that MaxSat-A$\nu(F)$ can be solved in expected time $O(8^{k+O(\sqrt{k})}(n+m)^{O(1)})$. Summarizing, we have the following:

Theorem 27. *There are algorithms solving* MaxSat-A$\nu(F)$ *in time* $O((2e)^{2k+O(\log^2 k)}(n+m)^{O(1)})$ *or expected time* $O(8^{k+O(\sqrt{k})}(n+m)^{O(1)})$.

7.6 Hardness of Kernelization

In this section, we show that MaxSat-A$\nu(F)$ does not have a polynomial-size kernel, unless coNP $\subseteq$ NP/poly. To do this, we use the concept of a *polynomial parameter transformation* [11, 32]: Let L and Q be parameterized problems. We say a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is a *polynomial parameter transformation* from L to Q if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for any $(x,k) \in \Sigma^* \times \mathbb{N}, (x,k) \in L$ if and only if $f(x,k) = (x',k') \in Q$, and $k' \leq p(k)$.

Lemma 33. [11, Theorem 3] *Let L and Q be parameterized problems, and suppose that L^c and Q^c are the derived classical problems*[1]*. Suppose that L^c is NP-complete, and $Q^c \in$ NP. Suppose that f is a polynomial parameter transformation from L to Q. Then, if Q has a polynomial-size kernel, then L has a polynomial-size kernel.*

Recall that whilst a decidable problem is FPT if an only if it has a kernel, the kernel obtain from the FPT algorithm is not polynomial-size. The next theorem shows MaxSat-A$\nu(F)$ has no polynomial-size kernel under a reasonable complexity-theoretic assumption. The proof is similar to the proof of Lemma 29.

Theorem 28. MaxSat-A$\nu(F)$ *has no polynomial-size kernel, unless coNP $\subseteq$ NP/poly.*

Proof. By [45, Theorem 3], there is no polynomial-size kernel for the problem of deciding whether a hypergraph H has a hitting set of size $|E(H)| - k$, where k is the parameter unless coNP $\subseteq$

[1]The parameters of L and Q are no longer parameters in L^c and Q^c; they are part of input.

NP/poly. We prove the theorem by a polynomial parameter reduction from this problem. Then the theorem follows from Lemma 33, as MaxSat-A$\nu(F)$ is NP-complete.

Given a hypergraph H on n vertices, construct a CNF formula F as follows. Let the variables of F be the vertices of H. For each variable x, let the unit clause (x) be a clause in F. For every edge e in H, let c_e be the clause containing the literal $\bar{x}$ for every $x \in E$. Observe that F is matched, and that H has a hitting set of size $|E(H)| - k$ if and only if $\text{sat}(F) \geq n + k$. □

Chapter 8

Unit-Conflict Free MaxSAT

8.1 Introduction

For a CNF formula F a pair of clauses are conflicting if they can not both be simultaneously satisfied. The *unit-conflict free (UCF)* form of a CNF formula F is formed by deleting all pairs of conflicting clauses. If F is UCF, then Lieberherr and Specker [79] proved that $\text{sat}(F) \geq \hat{\varphi}m$, where $\hat{\varphi} = (\sqrt{5}-1)/2$.

In this chapter we consider the following parameterization of MaxSAT, introduced by Mahajan and Raman [85] since $\hat{\varphi}m$ is an asymptotically tight lower bound for UCF CNF formulae:

MaxSat-UCF-A($\hat{\varphi}m$)
Instance: A UCF CNF formula F with m clauses.
Parameter: A nonnegative integer k.
Question: Decide whether $\text{sat}(F) \geq \hat{\varphi}m + k$.

Mahajan and Raman conjectured that MaxSat-UCF-A($\hat{\varphi}m$) is fixed-parameter tractable. To solve the conjecture in the affirmative, we show the existence of an $O(k)$-variable kernel for MaxSat-UCF-A($\hat{\varphi}m$) by obtaining an improvement of the Lieberherr-Specker lower bound.

A formula $F' = (V', C')$ is called a *subformula* of a CNF formula $F = (V, C)$ if $C' \subseteq C$ and V' is the set of variables in C'. If F' is a subformula of F then $F \setminus F'$ denotes the subformula obtained from F by deleting all clauses of F'. A formula $F = (V, C)$ is called *expanding* if for each $X \subseteq V$, the number of clauses containing at least one variable from X is at least $|X|$ [103]. It is known (this involves so-called matching autarkies, see Section 8.2 for details) that for each CNF formula $F = (V, C)$ a subformula $F' = (V', C')$ can be found in polynomial time such that $\text{sat}(F) = \text{sat}(F \setminus F') + |C'|$ and the subformula $F \setminus F'$ is expanding. The main technical result of this chapter is that $\text{sat}(F) \geq \hat{\varphi}|C| + (2 - 3\hat{\varphi})|V|/2$ for every expanding UCF CNF formula

$F = (V, C)$. Combining this inequality with the previous equality for $\mathrm{sat}(F)$, we conclude that for each UCF CNF formula $F = (V, C)$ a subformula $F' = (V', C')$ can be found in polynomial time such that

$$\mathrm{sat}(F) \geq \hat{\varphi}|C| + (1 - \hat{\varphi})|C'| + (2 - 3\hat{\varphi})|V \setminus V'|/2.$$

The last inequality improves the Lieberherr-Specker lower bound on $\mathrm{sat}(F)$.

The rest of this chapter is organized as follows. In Section 8.2, we give further terminology and notation and some basic results. Section 8.3 proves the improvement of the Lieberherr-Specker lower bound on $\mathrm{sat}(F)$ assuming correctness of the following lemma: if $F = (V, C)$ is a compact CNF formula, then $\mathrm{sat}(F) \geq \hat{\varphi}|C| + (2 - 3\hat{\varphi})|V|/2$ (we give definition of a compact CNF formula in the next section). We prove this non-trivial lemma in Section 8.4. In Section 8.5 we solve the conjecture of Mahajan and Raman [85] in the affirmative and improve their result on SAT-A$(m/2)$. We conclude the chapter with discussions and open problems.

8.2 Additional Terminology, Notation and Basic Results

We let $F = (V, C)$ denote a CNF formula with a set of variables V and a multiset of clauses C. It is normally assumed that each clause may appear multiple times in C. For the sake of convenience, we assume that each clause appears at most once, but allow each clause to have an integer *weight*. (Thus, instead of saying a clause c appears t times, we will say that c has weight t). If at any point a particular clause c appears more than once in C, we replace all occurrences of c with a single occurrence of the same total weight. We use $w(c)$ to denote the weight of a clause c. For any clause $c \notin C$ we set $w(c) = 0$. If $C' \subseteq C$ is a subset of clauses, then $w(C')$ denotes the sum of the weights of the clauses in C'. For a formula $F = (V, C)$ we will often write $w(F)$ instead of $w(C)$.

For a formula $F = (V, C)$ and a subset $U \subseteq V$ of variables, F_U denotes the subformula of F obtained from F by deleting all clauses without variables in U.

For a CNF formula $F = (V, C)$, a *truth assignment* is a function $\alpha : V \to \{\textsc{true}, \textsc{false}\}$. A truth assignment α *satisfies* a clause c if there exists $x \in V$ such that $x \in c$ and $\alpha(x) = \textsc{true}$, or $\bar{x} \in c$ and $\alpha(x) = \textsc{false}$. The *weight* of a truth assignment is the sum of the weights of all clauses satisfied by the assignment. The maximum weight of a truth assignment for F is denoted by $\mathrm{sat}(F)$.

A function $\beta : U \to \{\textsc{true}, \textsc{false}\}$, where U is a subset of V is called a *partial truth assignment*. A partial truth assignment $\beta : U \to \{\textsc{true}, \textsc{false}\}$ is an *autarky* if β satisfies all clauses of F_U. Autarkies are of interest, in particular, due to the following simple fact whose trivial proof is omitted.

Lemma 34. *Let $\beta : U \to \{\textsc{true}, \textsc{false}\}$ be an autarky for a CNF formula F. Then $\mathrm{sat}(F) = w(F_U) + \mathrm{sat}(F \setminus F_U)$.*

A version of Lemma 34 can be traced back to Monien and Speckenmeyer [88].

Recall that a formula $F = (V, C)$ is called *expanding* if $|X| \leq w(F_X)$ for each $X \subseteq V$. We associate a bipartite graph with a CNF formula $F = (V, C)$ as follows: the *bipartite graph* B_F *of* F has partite sets V and C and the edge vc is in B_F if and only if the variable v or its negation $\bar{v}$ appears in the clause c. Later we will make use of the following result which is a version of Hall's Theorem on matchings in bipartite graphs (cf. [107]).

Lemma 35. *The bipartite graph B_F has a matching covering V if and only if F is expanding.*

We call a CNF formula $F = (V, C)$ *compact* if the following conditions hold:

1. All clauses in F have the form (x) or $(\bar{x} \vee \bar{y})$ for some $x, y \in V$.
2. For every variable $x \in V$, the clause (x) is in C.

8.3 New Lower Bound for $\mathrm{sat}(F)$

We would like to prove a lower bound on $\mathrm{sat}(F)$ that has a term dependent on the number of variables. It is clear that for general CNF formula F such a bound is impossible. For consider a formula containing a single clause c containing a large number of variables. We can arbitrarily increase the number of variables in the formula, and the maximum number of satisfiable clauses will always be 1. We therefore need a reduction rule that cuts out 'excess' variables. Our reduction rule is based on the following lemma proved in Fleischner *et al.* [39] (Lemma 10) and Szeider [103] (Lemma 9).

Lemma 36. *Let $F = (V, C)$ be a CNF formula. Given a maximum matching in the bipartite graph B_F, in time $O(|C|)$ we can find an autarky $\beta : U \to \{\text{TRUE}, \text{FALSE}\}$ such that $|X| + 1 \leq w(F_X)$ for every $X \subseteq V \backslash U$.*

Note that the autarky found in Lemma 36 can be empty, i.e., $U = \emptyset$. An autarky found by the algorithm of Lemma 36 is of a special kind, called a matching autarky; such autarkies were used first by Aharoni and Linial [2]. Results similar to Lemma 36 have been obtained in the parameterized complexity literature as well, see, e.g., [81].

Lemmas 34 and 36 immediately imply the following:

Theorem 29. *[39, 103] Let F be a CNF formula and let $\beta : U \to \{\text{TRUE}, \text{FALSE}\}$ be an autarky found by the algorithm of Lemma 34. Then $\mathrm{sat}(F) = \mathrm{sat}(F \setminus F_U) + w(F_U)$ and $F \setminus F_U$ is an expanding formula.*

Our improvement of the Lieberherr-Specker lower bound on $\mathrm{sat}(F)$ for a UCF CNF formula F will follow immediately from Theorems 29 and 30 (stated below). It is much harder to prove Theorem 30 than Theorem 29, and our proof of Theorem 30 is based on the following quite non-trivial lemma that will be proved in the next section.

Lemma 37. *If $F = (V, C)$ is a compact CNF formula, then there exists a truth assignment with weight at least*

$$\hat{\varphi} w(C) + \frac{|V|(2 - 3\hat{\varphi})}{2},$$

where $\hat{\varphi} = (\sqrt{5} - 1)/2$, and such an assignment can be found in polynomial time.

The next proof builds on some of the basic ideas in [79].

Theorem 30. *If $F = (V, C)$ is an expanding UCF CNF formula, then there exists a truth assignment with weight at least*

$$\hat{\varphi} w(C) + \frac{|V|(2 - 3\hat{\varphi})}{2},$$

where $\hat{\varphi} = (\sqrt{5} - 1)/2$ and such an assignment can be found in polynomial time.

Proof. We will describe a polynomial-time transformation from F to a compact CNF formula F', such that $|V'| = |V|$ and $w(C') = w(C)$, and any truth assignment for F' can be turned into truth assignment for F of greater or equal weight. The theorem then follows from Lemma 37.

By Lemma 35, there is a matching in the bipartite graph B_F covering V. For each $x \in V$ let c_x be the unique clause associated with x in this matching. For each variable x, if the unit clause (x) or $(\bar{x})$ appears in C, leave c_x as it is for now. Otherwise, remove all variables except x from c_x. We now have that for every x, exactly one of (x), $(\bar{x})$ appears in C.

If $(\bar{x})$ is in C, replace every occurrence of the literal $\bar{x}$ in the clauses of C with x, and replace every occurrence of x with $\bar{x}$. We now have that Condition 2 in the definition of a compact formula is satisfied. For any clause c which contains more than one variable and at least one positive literal, remove all variables except one that occurs as a positive. For any clause which contains only negative literals, remove all but two variables. We now have that Condition 1 is satisfied. This completes the transformation.

In the transformation, no clauses or variables were completely removed, so $|V'| = |V|$ and $w(C') = w(C)$. Observe that the transformation takes polynomial time, and that any truth assignment for the compact formula F' can be turned into a truth assignment for F of greater or equal weight. Indeed, for some truth assignment for F', flip the assignment to x if and only if we replaced occurrences of x with $\bar{x}$ in the transformation. This gives a truth assignment for F such that every clause will be satisfied if its corresponding clause in F' is satisfied. □

Our main result follows immediately from Theorems 29 and 30.

Theorem 31. *Every UCF CNF formula $F = (V, C)$ contains a (possibly empty) subformula $F' = (V', C')$ that can be found in polynomial time and such that*

$$\text{sat}(F) \geq \hat{\varphi} w(C) + (1 - \hat{\varphi}) w(C') + (2 - 3\hat{\varphi})|V \setminus V'|/2.$$

8.4 Proof of Lemma 37

In this section, we use the fact that $\hat{\varphi} = (\sqrt{5}-1)/2$ is the positive root of the polynomial $\hat{\varphi}^2+\hat{\varphi}-1$. We call a clause $(\bar{x} \vee \bar{y})$ *good* if for every literal $\bar{z}$, the set of clauses containing $\bar{z}$ is not equal to $\{(\bar{x} \vee \bar{z}), (\bar{y} \vee \bar{z})\}$. We define $w_v(x)$ to be the total weight of all clauses containing the literal x, and $w_v(\bar{x})$ the total weight of all clauses containing the literal $\bar{x}$. (Note that $w_v(\bar{x})$ is different from $w(\bar{x})$, which is the weight of the particular clause $(\bar{x})$.) Let $\epsilon(x) = w_v(x) - \hat{\varphi} w_v(\bar{x})$. Let $\gamma = (2-3\hat{\varphi})/2 = (1-\hat{\varphi})^2/2$ and let $\Delta(F) = \mathrm{sat}(F) - \hat{\varphi} w(C)$.

To prove Lemma 37, we will use an algorithm, Algorithm $\mathcal{C}$, described below. We will show that, for any compact CNF formula $F = (V, C)$, Algorithm $\mathcal{C}$ finds a truth assignment with weight at least $\hat{\varphi} w(C) + \gamma |V|$. Step 3 of the algorithm removes any clauses which are satisfied or falsified by the given assignment of truth values to the variables. The purpose of Step 4 is to make sure the new formula is compact.

Algorithm $\mathcal{C}$ works as follows. Let F be a compact CNF formula. If F contains a variable x such that we can assign x TRUE and increase sufficiently the average number of satisfied clauses, we do just that (see Cases A and B of the algorithm). Otherwise, to achieve similar effect we have to assign truth values to two or three variables (see Cases C and D). Step 3 of the algorithm removes any clauses which are satisfied or falsified by the given assignment of truth values to the variables. The purpose of Step 4 is to make sure the new formula is compact.

ALGORITHM $\mathcal{C}$

While $|V| > 0$, repeat the following steps:

1. For each $x \in V$, calculate $w_v(x)$ and $w_v(\bar{x})$.
2. Mark some of the variables as TRUE or FALSE, according to the following cases:

   Case A: *There exists $x \in V$ with $w_v(x) \geq w_v(\bar{x})$.* Pick one such x and assign it TRUE.

   Case B: *Case A is false, and there exists $x \in V$ with $(1-\hat{\varphi})\epsilon(x) \geq \gamma$.* Pick one such x and assign it TRUE.

   Case C: *Cases A and B are false, and there exists a good clause.* Pick such a good clause $(\bar{x} \vee \bar{y})$, with (without loss of generality) $\epsilon(x) \geq \epsilon(y)$, and assign y FALSE and x TRUE.

   Case D: *Cases A, B and C are false.* Pick any clause $(\bar{x} \vee \bar{y})$ and pick z such that both clauses $(\bar{x} \vee \bar{z})$ and $(\bar{y} \vee \bar{z})$ exist. Consider the six clauses $(x), (y), (z), (\bar{x} \vee \bar{y}), (\bar{x} \vee \bar{z}), (\bar{y} \vee \bar{z})$ and all 2^3 assignments to the variables x, y, z, and pick an assignment maximizing the total weight of satisfied clauses among the six clauses.

3. Perform the following simplification: For any variable x assigned FALSE, remove any clause containing $\bar{x}$, remove the unit clause (x), and remove x from V. For any variable x assigned TRUE, remove the unit clause (x), remove $\bar{x}$ from any clause containing $\bar{x}$ and remove x from V.
4. For each y remaining, if there is a clause of the form $(\bar{y})$, do the following: If the weight of this clause is greater than $w_v(y)$, then replace all clauses containing the variable y (that is, literals y or $\bar{y}$) with one clause (y) of weight $w_v(\bar{y}) - w_v(y)$. Otherwise remove $(\bar{y})$ from C and change the weight of (y) to $w(y) - w(\bar{y})$.

In order to show that the algorithm finds a truth assignment with weight at least $\hat{\varphi}w(C) + \frac{|V|(2-3\hat{\varphi})}{2}$, we need the following two lemmas.

Lemma 38. *For a formula F, if we assign a variable x TRUE, and run Steps 3 and 4 of the algorithm, the resulting formula $F^* = (V^*, C^*)$ satisfies*

$$\Delta(F) \geq \Delta(F^*) + (1-\hat{\varphi})\epsilon(x).$$

Furthermore, we have $|V^*| = |V| - 1$, unless there exists $y \in V^*$ such that (y) and $(\bar{x} \vee \bar{y})$ are the only clauses containing y and they have the same weight. In this case, y is removed from V^*.

Proof. Observe that at Step 3, the clause (x) (of weight $w_v(x)$) is removed, clauses of the form $(\bar{x} \vee \bar{y})$ (total weight $w_v(\bar{x})$) become $(\bar{y})$, and the variable x is removed from V.

At Step 4, observe that for each y such that $(\bar{y})$ is now a clause, $w(C)$ is decreased by $2w_y$ and $\mathrm{sat}(F)$ is decreased by w_y, where $w_y = \min\{w(y), w(\bar{y})\}$. Let $q = \sum_y w_y$, and observe that $q \leq w_v(\bar{x})$. A variable y will only be removed at this stage if the clause $(\bar{x} \vee \bar{y})$ was originally in C. We therefore have

1. $\mathrm{sat}(F^*) \leq \mathrm{sat}(F) - w_v(x) - q$
2. $w(C^*) = w(C) - w_v(x) - 2q$

Using the above, we get

$$\begin{aligned}
\Delta(F) &= \mathrm{sat}(F) - \hat{\varphi} \cdot w(C) \\
&\geq (w_v(x) + \mathrm{sat}(F^*) + q) - \hat{\varphi}(w(C^*) + 2q + w_v(x)) \\
&= \Delta(F^*) + (1 - \hat{\varphi})w_v(x) - (2\hat{\varphi} - 1)q \\
&\geq \Delta(F^*) + (1 - \hat{\varphi})(\epsilon(x) + \hat{\varphi} \cdot w_v(\bar{x})) - (2\hat{\varphi} - 1)w_v(\bar{x}) \\
&= \Delta(F^*) + (1 - \hat{\varphi} - \hat{\varphi}^2)w_v(\bar{x}) + (1 - \hat{\varphi})\epsilon(x) \\
&= \Delta(F^*) + (1 - \hat{\varphi})\epsilon(x).
\end{aligned}$$

□

Lemma 39. *For a formula F, if we assign a variable x* FALSE*, and run Steps 3 and 4 of the algorithm, the resulting formula $F^{**} = (V^{**}, C^{**})$ has $|V^{**}| = |V| - 1$ and satisfies $\Delta(F) \geq \Delta(F^{**}) - \hat{\varphi}\epsilon(x)$.*

Proof. Observe that at Step 3, every clause containing the variable x is removed, and no other clauses will be removed at Steps 3 and 4. Since the clause (y) appears for every other variable y, this implies that $|V^{**}| = |V| - 1$. We also have the following: $\mathrm{sat}(F^{**}) \leq \mathrm{sat}(F) - w_v(\bar{x})$ and $w(C^{**}) = w(C) - w_v(\bar{x}) - w_v(x)$. Thus,

$$\begin{aligned}
\Delta(F) &= \mathrm{sat}(F) - \hat{\varphi}w(C) \\
&\geq (w_v(\bar{x}) + \mathrm{sat}(F^{**})) - \hat{\varphi}(w(C^{**}) + w_v(\bar{x}) + w_v(x)) \\
&= \Delta(F^{**}) + (1 - \hat{\varphi})w_v(\bar{x}) - \hat{\varphi}w_v(x) \\
&= \Delta(F^{**}) + (1 - \hat{\varphi})w_v(\bar{x}) - \hat{\varphi}(\epsilon(x) + \hat{\varphi} \cdot w_v(\bar{x})) \\
&= \Delta(F^{**}) + (1 - \hat{\varphi} - \hat{\varphi}^2)w_v(\bar{x}) - \hat{\varphi}\epsilon(x) \\
&= \Delta(F^{**}) - \hat{\varphi}\epsilon(x).
\end{aligned}$$

□

Now we are ready to prove Lemma 37.

Proof of Lemma 37: We will show that Algorithm $\mathcal{C}$ finds a truth assignment with weight at least $\hat{\varphi}w(C) + \frac{|V|(2-3\hat{\varphi})}{2}$. Note that the inequality in the lemma can be reformulated as $\Delta(F) \geq \gamma|V|$.

Let F and $\hat{\varphi}$ be defined as in the statement of the lemma. Note that at each iteration of the algorithm, at least one variable is removed. Therefore, we will show the lemma by induction on $|V|$.

If $|V| = 0$ then we are done trivially and if $|V| = 1$ then we are done as $\text{sat}(F) = w(C) \geq \hat{\varphi}w(C)+\gamma$ (as $w(C) \geq 1$). So assume that $|V| \geq 2$.

For the induction step, let $F' = (V', C')$ be the formula resulting from F after running Steps 1-4 of the algorithm, and assume that $\Delta(F') \geq \gamma|V'|$. We show that $\Delta(F) \geq \gamma|V|$, by analyzing each possible case in Step 2 separately.

Case A: *$w_v(x) \geq w_v(\bar{x})$ for some $x \in V$.* In this case we let x be TRUE, which by Lemma 38 implies the following:

$$\begin{aligned}\Delta(F) &\geq \Delta(F') + (1-\hat{\varphi})\epsilon(x)\\ &= \Delta(F') + (1-\hat{\varphi})(w_v(x) - \hat{\varphi}w_v(\bar{x}))\\ &\geq \Delta(F') + (1-\hat{\varphi})(w_v(x) - \hat{\varphi}w_v(x))\\ &= \Delta(F') + (1-\hat{\varphi})^2 w_v(x)\\ &= \Delta(F') + 2\gamma w_v(x).\end{aligned}$$

If $y \in V \setminus V'$, then either $y = x$ or $(\bar{x} \vee \bar{y}) \in C$. Therefore $|V| - |V'| \leq w_v(\bar{x}) + 1 \leq w_v(x) + 1$. As $w_v(x) \geq 1$ we note that $2\gamma w_v(x) \geq \gamma(w_v(x) + 1)$. This implies the following, by induction, which completes the proof of Case A.

$$\begin{aligned}\Delta(F) &\geq \Delta(F') + \gamma(w_v(x) + 1)\\ &\geq \gamma|V'| + \gamma(w_v(x) + 1) \geq \gamma|V|.\end{aligned}$$

Case B: *Case A is false, and $(1-\hat{\varphi})\epsilon(x) \geq \gamma$ for some $x \in V$.*

Again we let x be TRUE. Since $w_v(y) < w_v(\bar{y})$ for all $y \in V$, we have $|V| = |V'|+1$. Analogously to Case A, using Lemma 38, we get the following:

$$\begin{aligned}\Delta(F) &\geq \Delta(F') + (1-\hat{\varphi})\epsilon(x)\\ &\geq \gamma|V'| + \gamma = \gamma|V|.\end{aligned}$$

For Cases C and D, we generate a graph G from the set of clauses. The vertex set of G is the variables in V (i.e. $V(G) = V$) and there is an edge between x and y if and only if the clause $(\bar{x} \vee \bar{y})$ exists in C. A *good* edge in G is an edge $uv \in E(G)$ such that no vertex $z \in V$ has $N(z) = \{u, v\}$ (that is, an edge is good if and only if the corresponding clause is good).

Case C: *Cases A and B are false, and there exists a good clause $(\bar{x} \vee \bar{y})$.* Without loss of generality assume that $\epsilon(x) \geq \epsilon(y)$. We will first let y be FALSE and then we will let x be TRUE. By letting y be FALSE we get the following by Lemma 39, where F^{**} is defined in Lemma 39: $\Delta(F) \geq \Delta(F^{**}) - \hat{\varphi}\epsilon(x)$.

Note that the clause $(\bar{x} \vee \bar{y})$ has been removed so $w_v^{**}(\bar{x}) = w_v(\bar{x}) - w(\bar{x} \vee \bar{y})$ and $w_v^{**}(x) = w_v(x)$ (where $w_v^{**}(.)$ denote the weights in F^{**}). Therefore using Lemma 38 on F^{**} instead of F we get the following, where the formula F^* in Lemma 38 is denoted by F' below and $w^0 = w(\bar{x} \vee \bar{y})$:

$$\Delta(F^{**}) \geq \Delta(F') + (1-\hat{\varphi})(w_v(x) - \hat{\varphi}(w_v(\bar{x}) - w^0)).$$

First we show that $|V'| = |V^{**}| - 1 = |V| - 2$. Assume that $z \in V \setminus (V' \cup \{x, y\})$ and note that $N(z) \subseteq \{x, y\}$. Clearly $|N(z)| = 1$ as xy is a good edge. If $N(z) = \{y\}$ then $(z) \in C'$, so we must have $N(z) = \{x\}$. However the only way $z \notin V'$ is if $w_v(z) = w_v(\bar{z})$, a contradiction as Case A is false. Therefore, $|V'| = |V| - 2$, and the following holds by the induction hypothesis.

$$\begin{aligned}
\Delta(F) &\geq \Delta(F^{**}) - \hat{\varphi}\epsilon(x) \\
&\geq \Delta(F') + (1-\hat{\varphi})(w_v(x) - \hat{\varphi}(w_v(\bar{x}) - w^0)) - \hat{\varphi}\epsilon(x) \\
&\geq \gamma|V'| + (1-\hat{\varphi})(\epsilon(x) + \hat{\varphi}w^0) - \hat{\varphi}\epsilon(x) \\
&= \gamma|V| - 2\gamma + (1-\hat{\varphi})\hat{\varphi}w^0 - (2\hat{\varphi}-1)\epsilon(x).
\end{aligned}$$

We would be done if we can show that $2\gamma \leq (1-\hat{\varphi})\hat{\varphi}w^0 - (2\hat{\varphi}-1)\epsilon(x)$. As $w^0 \geq 1$ and we know that, since Case B does not hold, $(1-\hat{\varphi})\epsilon(x) < \gamma$, we would be done if we can show that $2\gamma \leq (1-\hat{\varphi})\hat{\varphi} - (2\hat{\varphi}-1)\gamma/(1-\hat{\varphi})$. This is equivalent to $\gamma = (1-\hat{\varphi})^2/2 \leq \hat{\varphi}(1-\hat{\varphi})^2$, which is true, completing the proof of Case C.

Case D: *Cases A, B and C are false.* Then G has no good edge.

Assume xy is some edge in G and $z \in V$ such that $N(z) = \{x, y\}$. As xz is not a good edge there exists a $v \in V$, such that $N(v) = \{x, z\}$. However v is adjacent to z and, thus, $v \in N(z) = \{x, y\}$, which implies that $v = y$. This shows that $N(y) = \{x, z\}$. Analogously we can show that $N(x) = \{y, z\}$. Therefore, the only clauses in C that contain a variable from $\{x, y, z\}$ form the following set: $S = \{(x), (y), (z), (\bar{x} \vee \bar{y}), (\bar{x} \vee \bar{z}), (\bar{y} \vee \bar{z})\}$.

Let F' be the formula obtained by deleting the variables x, y and z and all clauses containing them. Now consider the three assignments of truth values to x, y, z such that only one of the three variables is assigned FALSE. Observe that the total weight of clauses satisfied by these three assignments equals

$$w_v(\bar{x}) + w_v(\bar{y}) + w_v(\bar{z}) + 2(w(x) + w(y) + w(z)) = 2W,$$

where W is the total weight of the clauses in S. Thus, one of the three assignments satisfies the weight of at least $2W/3$ among the clauses in S. Observe also that $w(C) - w(C') \geq 6$, and, thus, the following holds.

$$\begin{aligned}
\Delta(F) &\geq 2(w(C) - w(C'))/3 + \mathrm{sat}(F') - \hat{\varphi}(w(C') + w(C) - w(C')) \\
&\geq \gamma|V'| + 2(w(C) - w(C'))/3 - \hat{\varphi}(w(C) - w(C')) \\
&= \gamma|V| - 3\gamma + (2 - 3\hat{\varphi})(w(C) - w(C'))/3 \\
&\geq \gamma|V| - 3\gamma + 2(2 - 3\hat{\varphi}) > \gamma|V|.
\end{aligned}$$

This completes the proof of the correctness of Algorithm $\mathcal{C}$. It remains to show that Algorithm $\mathcal{C}$ takes polynomial time.

Each iteration of the algorithm takes $O(nm)$ time. The algorithm stops when V is empty, and at each iteration some variables are removed from V. Therefore, the algorithm goes through at most n iterations and, in total, it takes $O(n^2m)$ time. This completes the proof of Lemma 37. □

Note that the bound $(2-3\hat{\varphi})/2$ in Lemma 37 cannot be improved due to the following example. Let l be any positive integer and let $F=(V,C)$ be defined such that $V=\{x_1,x_2,\ldots,x_l,y_1,y_2,\ldots,y_l\}$ and C contain the constraints (x_1), (x_2),..., (x_l), (y_1), (y_2),..., (y_l) and $(\bar{x}_1 \vee \bar{y}_1)$, $(\bar{x}_2 \vee \bar{y}_2)$,..., $(\bar{x}_l \vee \bar{y}_l)$. Let the weight of every constraint be one and note that for every i we can only satisfy two of the three constraints (x_i), (y_i) and $(\bar{x}_i \vee \bar{y}_i)$. Therefore $\mathrm{sat}(F) = 2l$ and the following holds:

$$\hat{\varphi}w(C) + \tfrac{|V|(2-3\hat{\varphi})}{2} = 3l\hat{\varphi} + \tfrac{2l(2-3\hat{\varphi})}{2} = l(3\hat{\varphi} + 2 - 3\hat{\varphi}) = 2l = \mathrm{sat}(F).$$

8.5 Parameterized Complexity Results

Recall that formulations of parameterized problems SAT-A$(m/2)$ and MaxSat-UCF-A$(\hat{\varphi}m)$ were given in Section 8.1.

Theorem 32. *The problem* MaxSat-UCF-A$(\hat{\varphi}m)$ *has a proper kernel with at most* $\lfloor(7+3\sqrt{5})k\rfloor$ *variables.*

Proof. Consider an instance $(F=(V,C),k)$ of the problem. By Theorem 29, there is an autarky $\beta : U \to \{\text{TRUE, FALSE}\}$ which can be found by the polynomial algorithm of Lemma 34 such that $\mathrm{sat}(F) = \mathrm{sat}(F \setminus F_U) + w(F_U)$ and $F \setminus F_U$ is an expanding formula.

If $U = V$, then $\mathrm{sat}(F) = w(F)$, and the kernel is trivial.

Now suppose that $U \neq V$ and denote $F \setminus F_U$ by $F' = (V', C')$. We want to choose an integral parameter k' such that (F,k) is a YES-instance of the problem if and only if (F',k') is a YES-instance of the problem. It is enough for k' to satisfy $\mathrm{sat}(F) - \lfloor\hat{\varphi}w(F)\rfloor - k = \mathrm{sat}(F') - \lfloor\hat{\varphi}w(F')\rfloor - k'$. By Theorem 29, $\mathrm{sat}(F') = \mathrm{sat}(F) - w(F) + w(F')$. Therefore, we can set $k' = k - w(F) + w(F') + \lfloor\hat{\varphi}w(F)\rfloor - \lfloor\hat{\varphi}w(F')\rfloor$. Since $w(F) - w(F') \geq \lceil\hat{\varphi}(w(F) - w(F'))\rceil \geq \lfloor\hat{\varphi}w(F)\rfloor - \lfloor\hat{\varphi}w(F')\rfloor$, we have $k' \leq k$.

By Theorem 30, if $k' \leq \frac{|V'|(2-3\hat{\varphi})}{2}$, then F is a YES-instance of the problem. Otherwise, $|V'| < \frac{2k}{2-3\hat{\varphi}} = (7+3\sqrt{5})k$. Note that F' is not necessarily a kernel as $w(F')$ is not necessarily bounded by a function of k. However, if $w(F') \geq 2^{2k/(2-3\hat{\varphi})}$ then we can solve the instance (F',k') in time $O(w(F')^2)$ and, thus, we may assume that $w(F') < 2^{2k/(2-3\hat{\varphi})}$, in which case, F' is the required kernel. □

Theorem 33. *The problem* SAT-A$(m/2)$ *has a proper kernel with at most* $4k$ *variables and* $(2\sqrt{5}+4)k \leq 8.473k$ *clauses.*

Proof. First, we reduce the instance to a UCF instance $F = (V,C)$. As in Theorem 32, in polynomial time, we can obtain an expanding formula $F' = (V', C')$. Again, we want to choose a parameter k' such that (F,k) is a YES-instance if and only if (F',k') is a YES-instance.

It is enough for k' to satisfy $\mathrm{sat}(F) - \lfloor w(F)/2\rfloor - k = \mathrm{sat}(F') - \lfloor w(F')/2\rfloor - k'$. By Theorem 29, $\mathrm{sat}(F') = \mathrm{sat}(F) - w(F) + w(F')$. Therefore, we can set $k' = k - \lceil w(F)/2\rceil + \lceil w(F')/2\rceil$. As $w(F') \leq w(F)$, we have $k' \leq k$.

By Theorem 30, there is a truth assignment for F' with weight at least $\hat{\varphi}w(F') + \frac{|V'|(2-3\hat{\varphi})}{2}$. Hence, if $k' \leq (\hat{\varphi} - 1/2)w(F') + \frac{|V'|(2-3\hat{\varphi})}{2}$, the instance is a YES-instance. Otherwise,

$$k' - \frac{|V'|(2-3\hat{\varphi})}{2} > (\hat{\varphi} - \frac{1}{2})w(F'). \tag{8.1}$$

The weaker bound $k' > (\hat{\varphi} - \frac{1}{2})w(F')$ is enough to give us the claimed bound on the total weight (i.e., the number) of clauses. To bound the number of variables, note that since F' is expanding, we can satisfy at least $|V'|$ clauses. Thus, if $w(F')/2 + k' \leq |V'|$, the instance is a YES-instance. Otherwise, $w(F')/2 + k' > |V'|$ and

$$2(\hat{\varphi} - \frac{1}{2})(|V'| - k') < (\hat{\varphi} - \frac{1}{2})w(F'). \tag{8.2}$$

Combining Inequalities (8.1) and (8.2), we obtain:

$$2(\hat{\varphi} - \frac{1}{2})(|V'| - k') < (\hat{\varphi} - \frac{1}{2})w(F') < k' - \frac{|V'|(2-3\hat{\varphi})}{2}.$$

This simplifies to $|V'| < 4k' \leq 4k$, giving the required kernel. □

Part III

Parameterizations above Poljak-Turzík Bound

Chapter 9

Acyclic Subgraph

9.1 Introduction

In this section we focus on the Acyclic Subgraph above Poljak-Turzík Bound problem:

Acyclic Subgraph above Poljak-Turzík Bound (ASAPT)
Instance: An oriented connected graph G with n vertices and m arcs.
Parameter: The integer k.
Question: Does G contain an acyclic subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + \frac{k}{4}$ arcs?

In doing so, we build on the method introduced in [29], showing that the methodology can be modified and applied to other families of graphs.

In a nutshell, the method uses both two-way reduction rules (i.e., rules reducing an instance to an equivalent one) and one-way reduction rules (in such a rule if the reduced instance is a Yes-instance, then the original instance is also a Yes-instance) to transform the input instance to a trivial graph. If the reduction rules do not allow us to conclude that the input instance is a Yes-instance, then the input instance has a relatively "regular" structure that can be used to solve the problem by a fixed-parameter dynamic programming algorithm. To establish the reduction rules and to show their "completeness", a structural result on undirected graphs is used, such as Lemma 47 in our case or Lemma 3 in [29].

Whilst our underlying approach is the same as [29], the proofs used are different, making use of the specific structure of each problem. In particular, the "regular" structure obtained here is rather different, and the dynamic programming algorithm and kernel proof are also completely different, other than the fact that in both papers the proofs are based on the "regular" structure of the graph. Finally, note that whilst the kernel obtained in [29] has $O(k^5)$ vertices, we obtain a kernel with just $O(k^2)$ vertices and $O(k^2)$ arcs.

In the next section, we obtain two basic results on oriented graphs. Two-way and one-way reduction rules are introduced in Sections 9.3 and 9.4, respectively. Fixed-parameter tractability of ASAPT is proved in Section 9.5. Section 9.6 is devoted to proving the existence of a polynomial kernel.

The maximum number of arcs in an acyclic subgraph of D will be denoted by $\mathrm{a}(D)$. Let $\gamma(D) = \frac{m}{2} + \frac{n-c}{4}$, where c is the number of connected components of D. By the Poljak-Turzík bound, we have

$$\mathrm{a}(G) \geq \gamma(G) \tag{9.1}$$

for every oriented graph G. A *tournament* is an oriented graph obtained from a complete graph by orienting its edges arbitrarily. A *directed p-cycle* is a directed cycle with p arcs.

9.2 Basic Results on Oriented Graphs

In our arguments we use the following simple correspondence between acyclic digraphs and orderings of vertices in digraphs. Let H be an acyclic spanning subgraph of a digraph D. It is well-known [7] and easy to see that there is an ordering $x_1, \ldots, x_n$ of vertices of D such that if $x_i x_j$ is an arc of H then $i < j$. On the other hand, any ordering $x_1, \ldots, x_n$ of vertices of a digraph $D = (V, A)$ leads to an acyclic spanning subgraph of D: consider the subgraph induced by $\{x_i x_j : x_i x_j \in A, i < j\}$. As we study maximum-size acyclic subgraphs, we may restrict ourselves to acyclic spanning subgraphs. Thus, we may use interchangeably the notions of acyclic spanning subgraphs and vertex orderings.

There are some known lower bounds on $\mathrm{a}(T)$ for tournaments T on n vertices, see, e.g., [101] and references therein. We show the following useful bound which we were unable to find in the literature.

Lemma 40. *For a tournament T on n vertices with $m = \binom{n}{2}$ arcs, we can, in polynomial time, find an acyclic subgraph with at least $\frac{m}{2} + \frac{3n}{4} - 1 = \gamma(T) + \frac{2n-3}{4}$ arcs, if n is even, or $\frac{m}{2} + \frac{3(n-1)}{4} - 1 = \gamma(T) + \frac{2n-6}{4}$ arcs, if n is odd.*

Proof. We prove the lemma by induction. The claim can easily be checked for $n = 1$ and $n = 2$ and we may assume that $n \geq 3$.

Consider first the case when n is even. Suppose that there exists a vertex x such that $d^+(x) \geq \frac{n}{2} + 1$. Consider the tournament $T' = T - x$, with $m' = m - (n-1)$ arcs and $n' = n - 1$ vertices. By induction, there is an ordering on T' that produces an acyclic spanning subgraph H' of T' such that

$$\mathrm{a}(H') \geq \frac{m'}{2} + \frac{3(n'-1)}{4} - 1 = \frac{m-(n-1)}{2} + \frac{3(n-2)}{4} - 1 = \frac{m}{2} + \frac{3n}{4} - \frac{n}{2} - 2.$$

Now add x to the beginning of this ordering. This produces an acyclic spanning subgraph H of T such that $\mathrm{a}(H) \geq \mathrm{a}(H') + \frac{n}{2} + 1 \geq \frac{m}{2} + \frac{3n}{4} - 1$.

If there is a vertex x such that $d^-(x) \geq \frac{n}{2} + 1$, the same argument applies, but x is added to the end of the ordering.

Otherwise, for every vertex x of T, $d^+(x) \in \{\frac{n}{2} - 1, \frac{n}{2}\}$. Moreover, by considering the sum of out-degrees, exactly half the vertices have out-degree $\frac{n}{2}$. Hence, if $n \geq 4$, there are at least two vertices with out-degree $\frac{n}{2}$. Let x and y be two such vertices, and suppose, without loss of generality, that there is an arc from x to y. Now consider $T' = T - \{x, y\}$ with $m' = m - (2n - 3)$ edges and $n' = n - 2$ vertices. By induction, there is an ordering on the vertices of T' that produces an acyclic subgraph with at least $\frac{m'}{2} + \frac{3n'}{4} - 1 = \frac{m}{2} + \frac{3n}{4} - n - 1$ arcs. Place x and y at the beginning of this ordering, with x occurring before y. Then this will add all the arcs from x and y to the acyclic subgraph. Thus, $\mathrm{a}(T) \geq \frac{m}{2} + \frac{3n}{4} - n - 1 + n = \frac{m}{2} + \frac{3n}{4} - 1$.

Now suppose that n is odd. Let x be any vertex in T, and let $T' = T - x$. By induction, there is an ordering on T' that produces an acyclic subgraph with at least $\frac{m'}{2} + \frac{3n'}{4} - 1$ arcs, where $n' = n - 1$ is the number of vertices and $m' = m - (n - 1)$ is the number of arcs in T'. By placing x either at the beginning or end of this ordering, we may add at least $(n-1)/2$ arcs. Thus, $\mathrm{a}(T) \geq \frac{m-(n-1)}{2} + \frac{3(n-1)}{4} - 1 + \frac{n-1}{2} = \frac{m}{2} + \frac{3(n-1)}{4} - 1$. □

Lemma 41. *Let S be a nonempty set of vertices of an oriented graph G such that both $G - S$ and $G[S]$ are connected. If $\mathrm{a}(G - S) \geq \gamma(G - S) + \frac{k'}{4}$ and $\mathrm{a}(G[S]) \geq \gamma(G[S]) + \frac{k''}{4}$, then $\mathrm{a}(G) \geq \gamma(G) + \frac{k'+k''-1}{4} + \frac{|d^+(S)-d^-(S)|}{2}$. In particular, $\mathrm{a}(G) \geq \gamma(G) + \frac{k'+k''-1}{4}$ if $|E(S, V(G) \setminus S)|$ is even and $\mathrm{a}(G) \geq \gamma(G) + \frac{k'+k''+1}{4}$, if $|E(S, V(G) \setminus S)|$ is odd.*

Proof. Form an acyclic subgraph on G as follows. Assume without loss of generality that $d^+(S) \geq d^-(S)$. Pick the arcs leaving S together with the arcs of the acyclic subgraphs in $G - S$ and $G[S]$. This forms an acyclic subgraph H. Let $m = m' + m'' + \bar{m}$ and $n = n' + n''$, where $G - S$ has m' arcs and n' vertices, $G[S]$ has m'' arcs and n'' vertices and $\bar{m} = d^+(S) + d^-(S)$. The acyclic subgraph H has at least $\gamma(G - S) + \frac{k'}{4} + \gamma(G[S]) + \frac{k''}{4} + \frac{\bar{m}}{2} + \frac{d^+(S)-d^-(S)}{2} = \frac{m'+m''+\bar{m}}{2} + \frac{n'-1}{4} + \frac{n''-1}{4} + \frac{k'}{4} + \frac{k''}{4} + \frac{d^+(S)-d^-(S)}{2} = \gamma(G) + \frac{k'+k''-1}{4} + \frac{d^+(S)-d^-(S)}{2}$ arcs, as required. □

9.3 Two-way Reduction Rules

In the rest of this chapter, G stands for an arbitrary connected oriented graph with n vertices and m arcs. We initially apply two 'two-way' reduction rules to (G, k) to form a new instance (G', k) such that (G', k) is a YES-instance of ASAPT if and only if (G, k) is a YES-instance of ASAPT (i.e., the value of the parameter remains unchanged). We denote the number of vertices and arcs in G' by n' and m', respectively.

Reduction Rule 9.1. *Let x be a vertex and S a set of two vertices such that $G[S]$ is a component of $G - x$ and $G[S \cup \{x\}]$ is a directed 3-cycle. Then $G' := G - S$.*

Lemma 42. *If (G', k) is an instance obtained from (G, k) by an application of Rule 9.1, then G' is connected, and (G', k) is a* YES*-instance of* ASAPT *if and only if (G, k) is a* YES*-instance of* ASAPT.

Proof. Any two components of $G' - x$ will be connected by x and so G' is connected. Since $\mathrm{a}(G') = \mathrm{a}(G) - 2$, $m' = m - 3$ and $n' = n - 2$, we have $\mathrm{a}(G) \geq \frac{m}{2} + \frac{n-1}{4} + \frac{k}{4}$ if and only if $\mathrm{a}(G') \geq \frac{m'}{2} + \frac{n'-1}{4} + \frac{k}{4}$. □

Reduction Rule 9.2. *Let a, b, c, d, e be five vertices in G such that $G[a, b, c]$ and $G[c, d, e]$ are directed 3-cycles, $G[a, b, c, d, e] = G[a, b, c] \cup G[c, d, e]$ and a, e are the only vertices in $\{a, b, c, d, e\}$ that are adjacent to a vertex in $G - \{a, b, c, d, e\}$. To obtain G' from G, delete b, c and d, add a new vertex x and three arcs such that $G[a, x, e]$ is a directed 3-cycle.*

Lemma 43. *If (G', k) is an instance obtained from (G, k) by an application of Rule 9.2, then G' is connected, and (G', k) is a* Yes*-instance of* ASAPT *if and only if (G, k) is a* Yes*-instance of* ASAPT*.*

Proof. Clearly, G' is connected. Note that $\mathrm{a}(G') = \mathrm{a}(G) - 2$, $m' = m - 3$ and $n' = n - 2$. Thus, we have $\mathrm{a}(G) \geq \frac{m}{2} + \frac{n-1}{4} + \frac{k}{4}$ if and only if $\mathrm{a}(G') \geq \frac{m'}{2} + \frac{n'-1}{4} + \frac{k}{4}$. □

9.4 One-way Reduction Rules

Recall that G stands for an arbitrary connected oriented graph with n vertices and m arcs. We will apply reduction rules transforming an instance (G, k) of ASAPT into a new instance (G', k'), where G' is an oriented graph with n' vertices and m' arcs, and k' is the new value of the parameter. We will see that for the reduction rules of this section the following property will hold: if (G', k') is a Yes-instance then (G, k) is a Yes-instance, but not necessarily vice versa. Thus, the rules of this section are called one-way reduction rules.

Reduction Rule 9.3. *Let x be a vertex such that $G - x$ is connected, and $d^+(x) \neq d^-(x)$. To obtain (G', k') remove x from G and reduce k by $2|d^+(x) - d^-(x)| - 1$.*

Lemma 44. *If (G', k') is an instance reduced from (G, k) by an application of Rule 9.3, then G' is connected, and if (G', k') is a* Yes*-instance then (G, k) is a* Yes*-instance.*

Proof. Let (G', k') be a Yes-instance. Then by Lemma 41 with $S = \{x\}$ and $k'' = 0$, $\mathrm{a}(G) \geq \gamma(G) + \frac{k'-1}{4} + \frac{|d^+(S) - d^-(S)|}{2} = \gamma(G) + \frac{k}{4}$, as required. □

Reduction Rule 9.4. *Let S be a set of vertices such that $G - S$ is connected, $G[S]$ is a tournament, and $|S| \geq 4$. To obtain (G', k'), remove S from G and reduce k by $2|S| - 4$ if S is even, or $2|S| - 7$ if $|S|$ is odd.*

Lemma 45. *If (G', k') is an instance obtained from (G, k) by an application of Rule 9.4, then G' is connected, and if (G', k') is a* Yes*-instance then (G, k) is a* Yes*-instance.*

Proof. Suppose $|S|$ is even. By Lemma 40, $\mathrm{a}(G[S]) \geq \gamma(G[S]) + \frac{2|S|-3}{4}$. By Lemma 41, if $\mathrm{a}(G') \geq \gamma(G') + (k - 2|S| + 4)/4$, then $\mathrm{a}(G) \geq \gamma(G) + \frac{(k-2|S|+4)+(2|S|-3)-1}{4} = \gamma(G) + \frac{k}{4}$, as required.

A similar argument applies in the case when $|S|$ is odd, except the bound from Lemma 40 is $\gamma(G[S]) + \frac{2|S|-6}{4}$, and so $k' = k - (2|S| - 7)$ is applied. □

Reduction Rule 9.5. *Let S be a set of three vertices such that the underlying graph of $G[S]$ is isomorphic to P_3, and $G - S$ is connected. To obtain (G', k'), remove S from G and reduce k by 1.*

Lemma 46. *If (G', k') is an instance obtained from (G, k) by an application of Rule 9.5, then G' is connected, and if (G', k') is a* YES*-instance then (G, k) is a* YES*-instance.*

Proof. Observe that $\mathrm{a}(G[S]) = \gamma(G[S]) + \frac{1}{2}$. Hence, by Lemma 41, if $\mathrm{a}(G') \geq \gamma(G') + (k-1)/4$, then $\mathrm{a}(G) \geq \gamma(G) + k/4$. □

9.5 Fixed-Parameter Tractability of ASAPT

The next lemma follows immediately from a nontrivial structural result of Crowston *et al.* (Lemma 3 in [29]).

Lemma 47. *Given any connected undirected graph H, at least one of the following properties holds:*

A *There exist $v \in V(H)$ and $X \subseteq V(H)$ such that X is a connected component of $H - v$ and X is a clique;*

B *There exist $a, b, c \in V(H)$ such that $H[\{a, b, c\}]$ is isomorphic to P_3 and $H - \{a, b, c\}$ is connected;*

C *There exist $x, y \in V(H)$ such that $\{x, y\} \notin E(H)$, $H - \{x, y\}$ is disconnected, and for all connected components X of $H - \{x, y\}$, except possibly one, $X \cup \{x\}$ and $X \cup \{y\}$ are cliques.*

Lemma 48. *For any connected oriented graph G with at least one edge, one of Rules 9.1, 9.3, 9.4, 9.5 applies.*

Proof. If there is a vertex $x \in X$ such that $G - x$ is connected and $d^+(x) \neq d^-(x)$ (we will call such a case an *unbalanced case*), then Rule 9.3 applies. Thus, assume that for each $x \in X$ such that $G - x$ is connected we have $d^+(x) = d^-(x)$.

Consider the case when property A holds. If $|X| \geq 4$, Rule 9.4 applies on $S = X$. If $|X| = 3$, there has to be exactly one arc between X and v and $G[X]$ is a directed 3-cycle as otherwise we have an unbalanced case. Let $x \in X$ be the endpoint of this arc in X. Then Rule 9.1 applies with $S = X \setminus \{x\}$. If $|X| = 2$, then $G[X \cup \{v\}]$ is a directed 3-cycle (as otherwise we have an unbalanced case) and so Rule 9.1 applies. We cannot have $|X| = 1$ as this is an unbalanced case.

If property B holds, then Rule 9.5 can be applied to the path P_3 formed by a, b, c in the underlying graph of G.

Consider the case when property C holds. We may assume without loss of generality that the non-tournament component is adjacent to y.

Consider the subcase when $G - \{x, y\}$ has two connected components, X_1 and X_2, that are tournaments. Let $x_1 \in X_1$, $x_2 \in X_2$ and observe that the subgraph induced by x_1, x, x_2 forms a P_3 in the underlying graph of G and $G - \{x_1, x, x_2\}$ is connected, and so Rule 9.5 applies.

Now consider the subcase when $G - \{x, y\}$ has only one connected component X that is a tournament. If $|X| \geq 3$, then $X \cup \{x\}$ is a tournament with least four vertices, and so Rule 9.4 applies. If $|X| = 2$, then let $X = \{a, b\}$. Observe that a is adjacent to three vertices, b, x, y, and so we have an unbalanced case to which Rule 9.3 applies. Finally, $X = \{a\}$ is a singleton, then observe that x, a, y form a P_3 in the underlying graph of G and $G - \{x, a, y\}$ is connected, and so Rule 9.5 applies. □

In this chapter, we consider the one-vertex undirected graph as 2-connected. A maximal 2-connected induced subgraph of an undirected graph is called a *block*. An undirected graph H is called a *forest of cliques* if each block of H is a clique. A subgraph B of an oriented graph G is a *block* if $\text{UN}(B)$ is a block in $\text{UN}(G)$. An oriented graph G is a *forest of cliques* if $\text{UN}(G)$ is a forest of cliques. A connected graph H that is a forest of cliques is known as a *tree of cliques*.

Lemma 49. *Given a connected oriented graph G and integer k, we can either show that (G, k) is a* YES*-instance of* ASAPT*, or find a set U of at most $3k$ vertices such that $G - U$ is a forest of cliques with the following properties:*

1. *Every block in $G - U$ contains at most three vertices;*
2. *Every block X in $G - U$ with $|X| = 3$ induces a directed 3-cycle in G;*
3. *Every connected component in $G - U$ has at most one block X with $|X| = 2$ vertices;*
4. *There is at most one block in $G - U$ with one vertex (i.e., there is at most one isolated vertex in $G - U$).*

Proof. Apply Rules 9.1, 9.3, 9.4, 9.5 exhaustively, and let U be the set of vertices removed by Rules 9.3, 9.4, and 9.5 (but not Rule 9.1). If we reduce to an instance (G'', k'') with $k'' \leq 0$, then by Lemmas 42, 44, 45 and 46, (G, k) is a YES-instance and we may return YES. Now assume that, in the completely reduced instance (G'', k''), $k'' > 0$. We will prove that $|U| \leq 3k$ and $G - U$ satisfies the four properties of the lemma.

Observe that each time k is decreased by a positive integer q, at most $3q$ vertices are added to U. Thus, $|U| \leq 3k$. The rest of our proof is by induction. Observe that, by Lemma 48, for the completely reduced instance (G'', k'') either $G'' = \emptyset$ or G'' consists of a single vertex. Thus, $G'' - U$ satisfies the four properties of the lemma, which forms the basis of our induction.

For the induction step, consider an instance (G'', k'') obtained from the previous instance (G', k') by the application of a reduction rule. By the induction hypothesis, $G'' - U$ satisfies the four properties of the lemma. In the application of each of Rules 9.3, 9.4 and 9.5, the vertices deleted are added to U. Hence $G'' - U = G' - U$ and we are done unless G'' is obtained from G' by an application of Rule 9.1. Recall that in Rule 9.1 we delete a set S such that $G[S \cup \{x\}]$ forms a directed 3-cycle. We do not add S to U. If $x \in G'' - U$, then in $G' - U$, $S \cup \{x\}$ forms a block of size 3 that is a directed 3-cycle. If $x \notin G'' - U$, then in $G' - U$, S forms a new connected component with one block S with $|S| = 2$ vertices. Thus, $G' - U$ satisfies the four properties. □

Theorem 34. *There is an algorithm for* ASAPT *of runtime* $O((3k)!n^{O(1)})$.

Proof. We may assume that for a connected oriented graph G we have the second alternative in the proof of Lemma 49, i.e., we are also given the set U of at most $3k$ vertices satisfying the four properties of Lemma 49. Consider an algorithm which generates all orderings of U, in time $O((3k)!)$ as $|U| \leq 3k$. An ordering $u_1, u_2, ..., u_{|U|}$ of U means that in the acyclic subgraph of G we are constructing, we keep only arcs of $G[U]$ of the form u_iu_j, $i < j$. For each ordering we perform the following polynomial-time dynamic programming procedure.

For each vertex $x \in G - U$, we define a vector $(x_0, \ldots, x_{t+1})$. Initially, set x_i to be the number of vertices $u_j \in U$ with an arc from u_j to x if $j \leq i$, or an arc from x to u_j if $i < j$. Note that x_i is the number of arcs between x and U in the acyclic subgraph under the assumption that in the ordering of the vertices of G, x is between u_i and u_{i+1}.

Given $v, w \in V(G - U)$ and an ordering of $U \cup \{v, w\}$, an arc vw is *satisfiable* if there is no u_p such that v is after u_p and w is before u_p, for some $p \in [|U|]$. Let T be a set of arcs and let $V(T)$ be the set of end-vertices of T. For an ordering $U \cup V(T)$, T is *satisfiable* if each arc is satisfiable, and the set T induces an acyclic subgraph.

If $G - U$ contains a block S that is itself a connected component, consider S and arbitrarily select a vertex x of S. Otherwise, find a block S in $G - U$ with only one vertex x adjacent to other vertices in $G - U$ (such a block exists as every block including an end-vertex of a longest path in $\mathrm{UN}(G) - U$ is such a block). Without loss of generality, assume that S has three vertices x, y, z (the case $|S| = 2$ can be considered similarly).

For each $i \in \{0, \ldots, t+1\}$, we let α_i be the maximum size of a set of satisfiable arcs between S and U under the restriction that x lies between u_i and u_{i+1}. Observe that $\alpha_i = \max_{j,h}(x_i + y_j + z_h + \beta(i, j, h))$, where $\beta(i, j, h)$ is the maximum size of a set of satisfiable arcs in $G[S]$ under the restriction that x lies between u_i and u_{i+1}, y lies between u_j and u_{j+1}, and z lies between u_h and u_{h+1}. Now delete $S \backslash \{x\}$ from G, and set $x_i = \alpha_i$ for each i.

Continue until each component of $G - U$ consists of a single vertex. Let x be such a single vertex, let G^* be the original graph G (i.e., given as input to our algorithm), and let X be the component of $G^* - U$ containing x. By construction, x_i is the maximum number of satisfiable arcs from arcs in X and arcs between X and U in G^*, under the assumption x is between u_i and u_{i+1}. Since each vertex x represents a separate component, the maximum acyclic subgraph in G has $Q + \sum_{x \in V(G-U)}(\max_i x_i)$ arcs, where Q is the number of arcs u_iu_j in $G[U]$ such that $i < j$.

Since the dynamic programming algorithm runs in time polynomial in n, running the algorithm for each permutation of U gives a runtime of $O((3k)!n^{O(1)})$. □

9.6 Polynomial Kernel

Lemma 50. *Let T be a directed 3-cycle, with vertices labeled 0 or 1. Then there exists an acyclic subgraph of T with two arcs, such that there is no arc from a vertex labeled 1 to a vertex labeled 0.*

Proof. Let $V(T) = \{a, b, c\}$ and assume that a, b are labeled 0. Since T is a cycle, either the arc ac or bc exists. This arc, together with the arc between a and b, form the required acyclic subgraph. A similar argument holds when two vertices in T are labeled 1. □

Recall that U was introduced in Lemma 49 as the set of vertices removed by Rules 9.3, 9.4, and 9.5. We say that a set $\{u, a, b\}$ of vertices is a *dangerous triangle* if $u \in U$, $G[a, b]$ is a block in $G - U$, and $G[u, a, b]$ is a directed 3-cycle.

Lemma 51. *For a vertex $u \in U$, let t_u denote the number of neighbors of u in $G - U$ which do not appear in a dangerous triangle containing u. If $t_u \geq 4k$, then we have a* YES*-instance.*

Proof. Let S denote the subgraph of $G - U$ consisting of all components C of $G - U$ which have a neighbor of u. For each component C of S, let $t_u(C)$ denote the number of neighbors of u in C which do not appear in a dangerous triangle containing u.

For each vertex $x \in G - U$, label it 0 if there exists an arc from x to u, or 1 if there is an arc from u to x. Recall from Lemma 49 each connected component in $G - U$ has at most one block $X = \{x, y\}$ with $|X| = 2$. If one vertex x is labeled, assign y the same label. Finally, assign label 1 to any remaining unlabeled vertices in $G - U$.

We will now construct an acyclic subgraph H' of $G - U$ such that there is no arc from a vertex labeled 1 to a vertex labeled 0. We then extend this to an acyclic subgraph H containing all the arcs between u and S.

Consider each block X in $G - U$. If $|X| = 3$, and X is a directed 3-cycle, then by Lemma 50 there is an acyclic subgraph of X with two arcs. Add this to H'. Now suppose $|X| = 2$, and let a, b be the vertices of X with an arc from a to b. If $G[X \cup \{u\}]$ is a dangerous triangle, then a is labeled 1 and b is labeled 0. In this case we do not include the arc ab in H'. However, H will include the two arcs between X and u, which do not count towards $t_u(C)$. If $G[X \cup \{u\}]$ is not a dangerous triangle, then we include the arc ab in the acyclic subgraph H'. Finally, let H be the acyclic subgraph formed by adding all arcs between u and S to H'.

Observe that for each component C of S, if $G[C \cup \{u\}]$ contains no dangerous triangle then H contains at least $\gamma(C)$ arcs in $G[C]$ (by the construction of H') and $t_u(C)$ arcs between C and u (since all arcs between S and u are in H), and $\gamma(C \cup \{u\}) := \gamma(G[C \cup \{u\}]) = \gamma(C) + \frac{t_u(C)}{2} + \frac{1}{4}$. So H contains at least $\gamma(C \cup \{u\}) + \frac{t_u(C)}{2} - \frac{1}{4}$ arcs. Since $G[C \cup \{u\}]$ contains no dangerous triangle but C is adjacent to u, $t_u(C) \geq 1$, and so H contains at least $\gamma(C \cup \{u\}) + \frac{t_u(C)}{4}$ arcs.

If $G[C \cup \{u\}]$ contains a dangerous triangle then H contains at least $\gamma(C) - \frac{3}{4}$ arcs in $G[C]$ (this can be seen by contracting the arc in C appearing in the dangerous triangle, and observing that in the resulting component C', H has at least $\gamma(C')$ arcs) and $t_u(C) + 2$ arcs between C and u, and $\gamma(C \cup \{u\}) = \gamma(C) + \frac{t_u(C)+2}{2} + \frac{1}{4}$. Thus, H contains at least $\gamma(C \cup \{u\}) + \frac{t_u(C)}{2}$ arcs.

Let $C_1, C_2, \ldots, C_q$ be the components of S. Observe that $\gamma(S \cup \{u\}) = \sum_{i=1}^{q} \gamma(C_i \cup \{u\})$. Then by combining the acyclic subgraphs for each $G[C_i \cup \{u\}]$, we have that $a(G[S \cup \{u\}]) \geq \sum_{i=1}^{q} (\gamma(C_i \cup \{u\}) + \frac{t_u(C_i)}{4}) = \gamma(S \cup \{u\}) + \frac{t_u}{4}$.

Finally, observe $G - S - u$ has at most $3k$ component, since each component must contain a vertex of U. By repeated application of Lemma 41, this implies there is an acyclic subgraph of G with at least $\gamma(G) + \frac{t_u - 3k}{4}$ arcs. Hence, if $t_u \geq 4k$, we have a YES-instance. □

Using the above lemma and the fact that $|U| \leq 3k$ (by Lemma 49), we have that unless (G, k) is a YES-instance, there are at most $12k^2$ vertices in $G - U$ that are adjacent to a vertex in U and do not appear in a dangerous triangle with that vertex.

Lemma 52. *Let s be the number of components in $G - U$ in which every neighbor x of a vertex $u \in U$ appears in a dangerous triangle together with u. If $s \geq k$, we have a* YES*-instance.*

Proof. By Lemma 49 such a component C_i contains at most one block of size 2. Since only blocks of size 2 can have vertices in dangerous triangles, only the vertices from this block in C_i may be adjacent to a vertex in U. But since G is reduced by Rule 9.1, component C_i must consist of only this block. Moreover, this block must appear in at least two dangerous triangles. Let a_i, b_i be the vertices of C_i, $i = 1, \ldots, s$ and let $C = \cup_{i=1}^{s} \{a_i, b_i\}$. Let $a_i b_i$ be an arc for each $i = 1, \ldots, s$ and note that every arc of G containing a_i (b_i, respectively) is either $a_i b_i$ or is from U to a_i (from b_i to U, respectively). Let δ_i be the number of dangerous triangles containing a_i and b_i; note that $\delta_i \geq 2$.

By (9.1), $G - C$ has an acyclic subgraph H with at least $\gamma(G - C)$ arcs. Observe that we can add to H all arcs entering each a_i and leaving each b_i, $i = 1, \ldots, s$, and obtain an acyclic subgraph H^* of G. We will prove that H^* contains enough arcs to show that (G, k) is a YES-instance. Observe that $G - C$ has at most $|U| \leq 3k$ components and $G[C]$ has $2s$ vertices and $2\sum_{i=1}^{s} \delta_i + s$ arcs, and recall that each $\delta_i \geq 2$. Thus, the number of arcs in H^* is at least

$$\begin{aligned} \gamma(G - C) + 2\sum_{i=1}^{s} \delta_i &\geq \frac{m - 2\sum_{i=1}^{s} \delta_i - s}{2} + \frac{n - 2s - 3k}{4} + 2\sum_{i=1}^{s} \delta_i \\ &\geq \gamma(G) + \sum_{i=1}^{s} \delta_i - s - \frac{3k}{4} \geq \gamma(G) + \frac{k}{4}. \end{aligned}$$

□

Let H be an undirected forest of cliques, where each block contains at most three vertices. A block B of H is called a *leaf-block* if there is at most one vertex of B belonging to another block of H. We denote the set of leaf-blocks of H by $\mathcal{L}(H)$. A block B of H is called a *path-block* if there is another block B' of H such that B and B' have a common vertex c which belongs only to these two blocks, at most one vertex of B belongs to a block other than B', and at most one vertex of B' belongs to a block other than B. We denote the set of path-blocks which are not leaf-blocks by $\mathcal{P}(H)$.

Lemma 53. *For a forest of cliques H, with each block of size at most three, if $l = |\mathcal{L}(H)|$ and $p = |\mathcal{P}(H)|$ then $|V(H)| \leq 8l + 2p$.*

Proof. We prove the claim by induction on the number of blocks in H. The case when H has only one block is trivial. Thus, we may assume that H has at least two blocks and H is connected. Let B be a leaf-block of H, and obtain subgraph H' by deleting the vertices of B not belonging to another block. Note that $|V(H)| \leq |V(H')| + 2$.

Assume that H' has a leaf-block B' which is not a leaf-block in H. Observe that $B' \in \mathcal{P}(H)$ and by induction $|V(H)| \leq 2 + 8l + 2(p-1) \leq 8l + 2p$.

Now assume that $|\mathcal{L}(H')| = l - 1$. Observe that removal of B from H may lead to a neighbour of B, B', becoming a path-block in H', together with at most two blocks neighbouring B'. Thus, at most three blocks may become path-blocks in H'. By the induction hypothesis, $|V(H')| \leq 8(l-1) + 2(p+3)$. Hence, $|V(H)| \leq 8(l-1) + 2(p+3) + 2 \leq 8l + 2p$. □

Theorem 35. Acyclic Subgraph above Poljak-Turzík Bound (ASAPT) *has a kernel with* $O(k^2)$ *vertices and* $O(k^2)$ *arcs.*

Proof. Consider an instance of (G^*, k) of ASAPT. Apply Rules 9.1 and 9.2 to obtain an instance (G, k) reduced by Rules 9.1 and 9.2.

Assume that (G, k) is reduced by Rules 9.1 and 9.2 and it is a No-instance.

Now we will apply all reduction rules but Rule 9.2. As a result, we will obtain the set U of vertices deleted in Rules 9.3, 9.4, and 9.5. By Lemma 49, $|U| \leq 3k$ and, by Lemma 51, each $u \in U$ has at most $4k$ neighbors that do not appear in a dangerous triangle with u. By Lemma 52, there are at most $2k$ vertices in $G - U$ that appear in a dangerous triangle with every neighbor in U (there are at most k components, and each component has two vertices). Hence the number of neighbors in $G - U$ of vertices of U is at most $4k|U| + 2k = 12k^2 + 2k$.

Now we will adopt the terminology and notation of Lemma 53 (we extend it from $\mathrm{UN}(G-U)$ to $G - U$ as we have done earlier). Consider a leaf-block B. Since G is reduced by Rules 9.1 and 9.3, B must contain a vertex v adjacent to U, and furthermore, v is not contained in any other block. Hence, $|\mathcal{L}(G-U)| \leq 12k^2 + 2k$.

Next, we observe that Rule 9.2 implies there do not exist two adjacent 3-vertex blocks $B = \{a, b, c\}$, $B' = \{c, d, e\}$ such that only a and e belong to other blocks, unless one of b, c, d has a neighbor in U. Observe that each connected component of $G - U$ contains at most one 2-vertex block, so there are at most $12k^2 + 2k$ 2-vertex path blocks. Each 2-vertex path block is adjacent to at most two 3-vertex path blocks. Hence, $|\mathcal{P}(G-U)| \leq 6(12k^2 + 2k)$. So, by Lemma 53, $|V(G-U)| \leq 8(12k^2 + 2k) + 2 \cdot 6(12k^2 + 2k) = O(k^2)$, and so $|V(G)| \leq O(k^2) + 3k = O(k^2)$.

Finally, we show G has $O(k^2)$ arcs. There are at most $|U|^2$ arcs in U. Between $G-U$ and U there are at most $(4k+2k)|U|$ arcs. Finally, observe that $G-U$ has at most $|V(G-U)| \leq 20(12k^2+2k)$ blocks, and each block contains at most 3 arcs. Hence, $|A(G)| \leq |U|^2 + 60(12k^2 + 2k) \leq 9k^2 + 60(12k^2 + 2k) = O(k^2)$.

Thus, either (G, k) is a Yes-instance, or (G, k) forms a kernel with $O(k^2)$ vertices and $O(k^2)$ arcs. □

Chapter 10

Signed MaxCut

10.1 Introduction

In this chapter we consider the problem of finding a balanced subgraph in a signed graph, parameterized above the Poljak-Turzík Bound:

Signed Max Cut ATLB
Instance: A connected signed graph G with n vertices and m edges.
Parameter: The integer k.
Question: Does G contain a balanced subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + \frac{k}{4}$ edges?

10.2 Terminology, Notation and Preliminaries

A cycle C in G is called *positive* (*negative*) if the number of negative edges in C is even (odd)[1]. The following characterization of balanced graphs is well-known.

Theorem 36. *[59] A signed graph G is balanced if and only if every cycle in G is positive.*

Let $G = (V, E)$ be a signed graph. For a subset W of V, the *W-switch* of G is the signed graph G_W obtained from G by changing the signs of the edges between W and $V \setminus W$. Note that a signed graph G is balanced if and only if there exists a subset W of V (W may coincide with V) such that G_W has no negative edges. Indeed, if G_W has no negative edges, G is $(W, V \setminus W)$-balanced. If G is (V_1, V_2)-balanced, then G_{V_1} has no negative edges.

Deciding whether a signed graph is balanced is polynomial-time solvable.

[1] To obtain the sign of C simply compute the product of the signs of its edges.

Theorem 37. *[44] Let $G = (V, E)$ be a signed graph. Deciding whether G is balanced is polynomial-time solvable. Moreover, if G is balanced then, in polynomial time, we can find a subset W of V such that G_W has no negative edges.*

For a signed graph G, $\beta(G)$ will denote the maximum number of edges in a balanced subgraph of G. Furthermore, for a signed graph $G = (V, E)$, $\mathsf{pt}(G)$ denotes the Poljak-Turzík bound: $\beta(G) \geq \mathsf{pt}(G)$. If G is connected, then $\mathsf{pt}(G) = \frac{|E(G)|}{2} + \frac{|V(G)|-1}{4}$, and if G has t components, then $\mathsf{pt}(G) = \frac{|E(G)|}{2} + \frac{|V(G)|-t}{4}$. It is possible to find, in polynomial time, a balanced subgraph of G of size at least $\mathsf{pt}(G)$ [96].

The following easy property will be very useful in later proofs. It follows from Theorem 36 by observing that for a signed graph the Poljak-Turzík bound does not depend on the signs of the edges and that, for any cycle in G, the sign of the cycle in G and in G_W is the same.

Corollary 4. *Let $G = (V, E)$ be a signed graph and let $W \subset V$. Then $\mathsf{pt}(G_W) = \mathsf{pt}(G)$ and $\beta(G_W) = \beta(G)$.*

For a vertex set X in a graph G, $G[X]$ denotes the subgraph of G induced by X. For disjoint vertex sets X, Y of graph G, $E(X, Y)$ denotes the set of edges between X and Y. A *bridge* in a graph is an edge that, if deleted, increases the number of connected components of the graph. A *block* of a graph is either a maximal 2-connected subgraph or a connected component containing only one vertex.

For an edge set F of a signed graph G, F^+ and F^- denote the set of positive and negative edges of F, respectively. For a signed graph $G = (V, E)$, the *dual* of G is the signed graph $\bar{G} = (V, \bar{E})$, where $\bar{E}^+ = E^-$ and $\bar{E}^- = E^+$. A cycle in G is *dually positive* (*dually negative*) if the same cycle in $\bar{G}$ is positive (negative).

For a graph $G = (V, E)$, the *neighborhood* $N_G(W)$ of $W \subseteq V$ is defined as $\{v \in V : vw \in E, w \in W\} \setminus W$; the vertices in $N_G(W)$ are called *neighbors of* W. If G is a signed graph, the *positive* neighbors of $W \subseteq V$ are the neighbors of W in $G^+ = (V, E^+)$; the set of positive neighbors is denoted $N_G^+(W)$. Similarly, for the *negative* neighbors and $N_G^-(W)$.

The next theorem is the 'dual' of Theorem 36, in the sense that it is its equivalent formulation on the dual of a graph.

Theorem 38. *Let $G = (V, E)$ be a signed graph. Then the dual graph $\bar{G}$ is balanced if and only if G does not contain a dually negative cycle.*

In the next sections, the notion of *forest of cliques* introduced in [29] plays a key role. A connected graph is a *tree of cliques* if the vertices of every cycle induce a clique. A *forest of cliques* is a graph whose components are trees of cliques. It follows from the definition that in a forest of cliques any block is a clique.

Note that a forest of cliques is a *chordal* graph, i.e., a graph in which every cycle has a chord, that is an edge between two vertices which are not adjacent in the cycle. The next lemma is a characterization of chordal graphs which have a balanced dual. A *triangle* is a cycle with three edges.

Corollary 5. *Let $G = (V, E)$ be a signed chordal graph. Then $\bar{G}$ is balanced if and only if G does not contain a positive triangle.*

Proof. If G contains a positive triangle, then, by Theorem 38, $\bar{G}$ is not balanced.

Now suppose that G is not balanced. By Theorem 38, G contains a dually negative cycle, i.e., a cycle with odd number of positive edges, but all triangles in G are negative by hypothesis. Let $C = v_1v_2 \dots v_lv_1$ be a dually negative cycle of minimum length and note that $l > 3$ as a dually negative triangle is positive. Since the graph is chordal, we can find three consecutive vertices of C that form a triangle T. Suppose $T = v_1v_2v_3v_1$. Recall that T is negative. So, if both v_1v_2 and v_2v_3 are positive edges (or negative edges), then v_1v_3 must be a negative edge; otherwise if one of the two edges is positive and the other negative, then v_1v_3 is a positive edge. In both cases, we conclude that C contains an odd number of positive edges if and only if $C' = v_1v_3v_4 \dots v_lv_1$ does, which is a contradiction since we supposed l to be the minimum length of a dually negative cycle. □

Corollary 6. *Let $(G = (V, E), k)$ be an instance $\mathcal{I}$ of* Signed Max Cut ATLB*, let $X \subseteq V(G)$ and let $G[X]$ be a chordal graph which does not contain a positive triangle. Then there exists a set $W \subseteq X$, such that $\widetilde{\mathcal{I}} = (G_W, k)$ is equivalent to $\mathcal{I}$, and $G_W[X]$ does not contain positive edges.*

Proof. By Corollary 5, $\bar{G}[X]$ is balanced: hence, by definition of balanced graph, there exists $W \subseteq X$ such that $\bar{G}_W[X]$ contains only positive edges, which means that $G_W[X]$ contains only negative edges. By Corollary 4, (G_W, k) is an instance equivalent to the original one. □

Lastly, the next lemmas describe useful properties of Max Cut ATLB which still hold for Signed Max Cut ATLB.

Lemma 54. *Let $G = (V, E)$ be a connected signed graph and let $V = U \cup W$ such that $U \cap W = \emptyset$, $U \neq \emptyset$ and $W \neq \emptyset$. Then $\beta(G) \geq \beta(G[U]) + \beta(G[W]) + \frac{1}{2}|E(U, W)|$. In addition, if $G[U]$ has c_1 components, $G[W]$ has c_2 components, $\beta(G[U]) \geq \mathsf{pt}(G[U]) + \frac{k_1}{4}$ and $\beta(G[W]) \geq \mathsf{pt}(G[W]) + \frac{k_2}{4}$, then $\beta(G) \geq \mathsf{pt}(G) + \frac{k_1+k_2-(c_1+c_2-1)}{4}$.*

Proof. Let H (F) be a balanced subgraph of $G[U]$ ($G[W]$) with maximum number of edges and let H (F) be (U_1, U_2)-balanced ((W_1, W_2)-balanced). Let $E_1 = E^+(U_1, W_1) \cup E^+(U_2, W_2) \cup E^-(U_1, W_2) \cup E^-(U_2, W_1)$ and $E_2 = E(U, W) \setminus E_1$. Observe that both $E(H) \cup E(F) \cup E_1$ and $E(H) \cup E(F) \cup E_2$ induce balanced subgraphs of G and the largest of them has at least $\beta(G[U]) + \beta(G[W]) + \frac{1}{2}|E(U, W)|$ edges.

Now, observe that $\mathsf{pt}(G) = \mathsf{pt}(G[U]) + \mathsf{pt}(G[W]) + \frac{1}{2}|E(U, W)| + \frac{c_1+c_2-1}{4}$. Hence $\beta(G) \geq \mathsf{pt}(G) + \frac{k_1+k_2-(c_1+c_2-1)}{4}$. □

Lemma 55. *Let $G = (V, E)$ be a signed graph, $v \in V$ a cutvertex, Y a connected component of $G - v$ and $G' = G - Y$. Then $\mathsf{pt}(G) = \mathsf{pt}(G[V(Y) \cup \{v\}]) + \mathsf{pt}(G')$ and $\beta(G) = \beta(G[V(Y) \cup \{v\}]) + \beta(G')$.*

Proof. The first equality is easily verified. Concerning the other, let H_1 be a (V_1^1, V_2^1)-balanced subgraph of $G[V(Y) \cup \{v\}]$ of size $\beta(G[V(Y) \cup \{v\}])$ and H_2 be a (V_1^2, V_2^2)-balanced subgraph of G' of size $\beta(G')$. One may assume that $v \in V_1^i$ for $i = 1, 2$. Therefore the balanced subgraph H of G induced by $V_1 = V_1^1 \cup V_1^2$ and $V_2 = V_2^1 \cup V_2^2$ is of size $\beta(G[V(Y) \cup \{v\}]) + \beta(G')$, which means that $\beta(G) \geq \beta(G[V(Y) \cup \{v\}]) + \beta(G')$. On the other hand, any balanced subgraph H of G induces balanced subgraphs of $G[V(Y) \cup \{v\}]$ and G', which implies that $\beta(G) \leq \beta(G[V(Y) \cup \{v\}]) + \beta(G')$. □

10.3 Fixed-Parameter Tractability

In this section, we prove that Signed Max Cut ATLB is FPT by designing an algorithm of running time[2] $O^*(8^k)$. This algorithm is a generalization of the FPT algorithm obtained in [29] to solve Max Cut ATLB. Given an instance $(G = (V, E), k)$ of Max Cut ATLB, the algorithm presented in [29] applies some reduction rules that either answer Yes for Max Cut ATLB or produce a set S of at most $3k$ vertices such that $G - S$ is a forest of cliques.

A key idea of this section is that it is possible to extend these rules such that we include into S at least one vertex for every dually negative cycle of G. As a result, Theorem 38 ensures that solving Signed Max Cut ATLB on $G-S$ is equivalent to solving Max Cut ATLB. Therefore, it is possible to guess a partial solution on S and then solve Max-Cut-with-Weighted-Vertices[3] on $G - S$. Since a forest of cliques is a chordal graph, Corollary 5 implies that it is enough to put into S at least one vertex for every positive triangle in G (instead of every dually negative cycle). Our reduction rules are inspired by the rules used in [29], but our rules are more involved in order to deal with positive triangles.

The rules apply to an instance (G, k) of Signed Max Cut ATLB and output an instance (G', k') where G' is obtained by deleting some vertices of G. In addition, the rules can mark some of the deleted vertices: marked vertices will form the set S such that $G - S$ is a forest of cliques. Note that every time a rule marks some vertices, it also decreases the parameter k.

The instance (G', k') that the rules produce does not have to be equivalent to (G, k), but it has the property that if it is a Yes-instance, then (G, k) is a Yes-instance too. For this reason, these rules are called *one-way* reduction rules [20].

Note that in the description of the rules, G is a connected signed graph, and C and Y denote connected components of a signed graph such that C is a clique which does not contain a positive triangle.

Reduction Rule 10.1. *If $abca$ is a positive triangle such that $G - \{a, b, c\}$ is connected, then mark a, b, c, delete them and set $k' = k - 3$.*

[2] In the O^*-notation widely used in parameterized algorithmics, we omit not only constants, but also polynomial factors.

[3] This problem is defined just before Theorem 40.

Reduction Rule 10.2. *If $abca$ is a positive triangle such that $G - \{a, b, c\}$ has two connected components C and Y, then mark a, b, c, delete them, delete C, and set $k' = k - 2$.*

Reduction Rule 10.3. *Let C be a connected component of $G - v$ for some vertex $v \in G$. If there exist $a, b \in V(C)$ such that $G - \{a, b\}$ is connected and there is an edge av but no edge bv, then mark a and b, delete them and set $k' = k - 2$.*

Reduction Rule 10.4. *Let C be a connected component of $G - v$ for some vertex $v \in G$. If there exist $a, b \in C$ such that $G - \{a, b\}$ is connected and $vabv$ is a positive triangle, then mark a and b, delete them and set $k' = k - 4$.*

Reduction Rule 10.5. *If there is a vertex $v \in V(G)$ such that $G - v$ has a connected component C, $G[V(C) \cup \{v\}]$ is a clique in G, and $G[V(C) \cup \{v\}]$ does not contain a positive triangle, then delete C and set $k' = k$.*

Reduction Rule 10.6. *If $a, b, c \in V(G)$, $\{ab, bc\} \subseteq E(G)$ but $ac \notin E(G)$, and $G - \{a, b, c\}$ is connected, then mark a, b, c, delete them and set $k' = k - 1$.*

Reduction Rule 10.7. *Let C, Y be the connected components of $G - \{v, b\}$ for some vertices $v, b \in V(G)$ such that $vb \notin E(G)$. If $G[V(C) \cup \{v\}]$ and $G[V(C) \cup \{b\}]$ are cliques not containing any positive triangles, then mark v and b, delete them, delete C and set $k' = k - 1$.*

Definition 2. *A one-way reduction rule is* safe *if it does not transform a* No*-instance into a* Yes*-instance.*

The intuitive understanding of how a one-way reduction rule works is that it removes a portion of the graph (while decreasing the parameter from k to k') only if given any solution (i.e., a balanced subgraph) on the rest of the graph there is a way to extend it to the removed portion while always gaining an additional $k - k'$ over the Poljak-Turzík bound.

Lemma 56. *Let G be a connected graph. If C is a clique of G such that $G - C$ is connected and if C contains a positive triangle, then either Rule 10.1 or Rule 10.2 applies.*

Proof. Let $abca$ be a positive triangle in C. Suppose Rule 10.1 does not apply. This means that $G - \{a, b, c\}$ is not connected: more precisely, $G - \{a, b, c\}$ has two components $G - C$ and $C - \{a, b, c\}$. Note that $C - \{a, b, c\}$ cannot contain a positive triangle, or otherwise Rule 10.1 would have applied. Therefore, Rule 10.2 applies. □

Theorem 39. *Rules 10.1-10.7 are safe.*

Proof. **Rule 10.1:** Let $abca$ be a positive triangle as in the description of Rule 10.1. Suppose $\beta(G') \geq \mathsf{pt}(G') + \frac{k'}{4}$, where $k' = k - 3$. Since $abca$ is a positive triangle, by Lemma 54, we obtain $\beta(G) \geq \mathsf{pt}(G) + \frac{k'+3}{4} = \mathsf{pt}(G) + \frac{k}{4}$.

Rule 10.2: Let $abca$ be a positive triangle such that $G - \{a, b, c\}$ has two components C and Y. Suppose $\beta(Y) \geq \mathsf{pt}(Y) + \frac{k'}{4}$, where $k' = k - 2$. We know that $\beta(C) \geq \mathsf{pt}(C)$, and so by Lemma

54 we obtain $\beta(G[V(Y) \cup V(C)]) \geq \mathsf{pt}(G[V(Y) \cup V(C)]) + \frac{k'-1}{4}$. Since $abca$ is a positive triangle, using Lemma 54 again we obtain $\beta(G) \geq \mathsf{pt}(G) + \frac{k'+4-2}{4} = \mathsf{pt}(G) + \frac{k}{4}$.

Rule 10.3: Let v, a, b and C be as in the description of Rule 10.3. Assume there exists a (V_1', V_2')-balanced subgraph H' of G' with at least $\mathsf{pt}(G') + \frac{k'}{4}$ edges, where $k' = k-2$. By Corollary 6, we may assume that all edges in C are negative. In addition, we may assume that the edge av is negative (the other case is similar). Lastly, without loss of generality assume that $v \in V_1'$. Now, consider the balanced subgraph H of G induced by (V_1, V_2), where $V_1 = V_1' \cup \{b\}$ and $V_2 = V_2' \cup \{a\}$. Since $|E(a, V_1 \cap V(C)) \cup E(b, V_2 \cap V(C))| = |E(a, V_2 \cap V(C)) \cup E(b, V_1 \cap V(C))|$, it holds that $|E(H)| = |E(H')| + \frac{|E(\{a,b\}, V[C-\{a,b\}])|}{2} + 2$. Moreover, $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{|E(\{a,b\}, V[C-\{a,b\}])|}{2} + \frac{3}{2}$. Thus, $\beta(G) \geq \mathsf{pt}(G) + \frac{k'+2}{4} = \mathsf{pt}(G) + \frac{k}{4}$.

Rule 10.4: Let v, a, b and C be as in the description of Rule 10.4. Assume there exists a (V_1', V_2')-balanced subgraph H' of G' with at least $\mathsf{pt}(G') + \frac{k'}{4}$ edges, where $k' = k-4$. As in the proof for Rule 10.3, assume C only contains negative edges, the edge av is negative and $v \in V_1'$. Since $vabv$ is a positive triangle, this implies that the edge bv is positive.

Now, consider the balanced subgraph H of G induced by (V_1, V_2), where $V_1 = V_1' \cup \{b\}$ and $V_2 = V_2' \cup \{a\}$. As in the proof for Rule 10.3, it holds that $|E(a, V_1 \cap V(C)) \cup E(b, V_2 \cap V(C))| = |E(a, V_2 \cap V(C)) \cup E(b, V_1 \cap V(C))|$. Hence, $|E(H)| = |E(H')| + \frac{|E(\{a,b\}, V[C-\{a,b\}])|}{2} + 3$, while $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{|E(\{a,b\}, V[C-\{a,b\}])|}{2} + 2$. Thus, $\beta(G) \geq \mathsf{pt}(G) + \frac{k'+4}{4} = \mathsf{pt}(G) + \frac{k}{4}$.

Rule 10.5: Let v and C be as in the description of Rule 10.5. Suppose $\beta(G') \geq \mathsf{pt}(G') + \frac{k}{4}$. We know that $\beta(G[V(C) \cup \{v\}]) \geq \mathsf{pt}(G[V(C) \cup \{v\}])$. Then, by Lemma 55, $\beta(G) \geq \mathsf{pt}(G) + \frac{k}{4}$.

Rule 10.6: Let a, b, c be as in the description of Rule 10.6 and let $P = G[\{a, b, c\}]$. Note that $\mathsf{pt}(P) = \frac{2}{2} + \frac{3-1}{4} = \frac{3}{2}$ and, whatever the signs of its edges, P is a balanced graph by Theorem 36. Therefore, $\beta(P) = 2 = \mathsf{pt}(P) + \frac{1}{2}$. Suppose $\beta(G') \geq \mathsf{pt}(G') + \frac{k'}{4}$, where $k' = k-1$. Then by Lemma 54, $\beta(G) \geq \mathsf{pt}(G) + \frac{k'+2-1}{4} = \mathsf{pt}(G) + \frac{k}{4}$ edges.

Rule 10.7: Let v, b and C, Y be as in the description of Rule 10.7. Suppose $\beta(Y) \geq \mathsf{pt}(Y) + \frac{k'}{4}$, where $k' = k-1$. We claim that $\beta(G-Y) \geq \mathsf{pt}(G-Y) + \frac{1}{2}$. If this holds, using Lemma 54 we obtain that $\beta(G) \geq \mathsf{pt}(G) + \frac{k'+1}{4} = \mathsf{pt}(G) + \frac{k}{4}$.

Let $G'' = G - Y$ and $V(C) = \{v_1, \ldots, v_n\}$. Note that $\mathsf{pt}(G'') = \frac{n(n-1)}{4} + n + \frac{n+1}{4}$.

By Corollary 6 we may assume that G'' contains only negative edges. If n is even, consider the partition (V_1, V_2) of $V(G'')$ where $V_1 = \{v, b, v_1, \ldots, v_{\frac{n}{2}-1}\}$ and $V_2 = \{v_{\frac{n}{2}}, \ldots, v_n\}$. The balanced subgraph induced by this partition contains $(\frac{n}{2}+1)^2 = \mathsf{pt}(G'') + \frac{3}{4}$ edges. On the other hand, if n is odd, consider the partition (V_1, V_2) of $V(G'')$ where $V_1 = \{v, b, v_1, \ldots, v_{\frac{n-1}{2}}\}$ and $V_2 = \{v_{\frac{n+1}{2}}, \ldots, v_n\}$. The balanced subgraph induced by this partition contains $\frac{n+3}{2} \cdot \frac{n+1}{2} = \mathsf{pt}(G'') + \frac{2}{4}$ edges. □

We now show that the reduction rules preserve connectedness and that there is always one of them which applies to a graph with at least one edge. To show this, we use the following lemma, based on a result in [29] but first expressed in the following form in [20].

Lemma 57. *[20] For any connected graph Q, at least one of the following properties holds:*

A *There exist $v \in V(Q)$ and $X \subseteq V(Q)$ such that $G[X]$ is a connected component of $Q - v$ and $G[X]$ is a clique;*

B *There exist $a, b, c \in V(Q)$ such that $Q[\{a, b, c\}]$ is isomorphic to path P_3 and $Q - \{a, b, c\}$ is connected;*

C *There exist $v, b \in V(Q)$ such that $vb \notin E(Q)$, $Q - \{v, b\}$ is disconnected, and for all connected components $G[X]$ of $Q - \{v, b\}$, except possibly one, $G[X \cup \{v\}]$ and $G[X \cup \{b\}]$ are cliques.*

Lemma 58. *For a connected graph G with at least one edge, at least one of Rules 10.1-10.7 applies. In addition, the graph G' which is produced is connected.*

Proof. It is not difficult to see that the graph G' is connected, since when it is not obvious, its connectedness is part of the conditions for the rule to apply.

If property A of Lemma 57 holds, and $G[X]$ contains a positive triangle $abca$, then by Lemma 56 either Rule 10.1 or Rule 10.2 applies. If $2 \leq |N_G(v) \cap X| \leq |X| - 1$, then Rule 10.3 applies. If $|N_G(v) \cap X| = |X|$ and there exist $a, b \in X$ such that $vabv$ is a positive triangle, Rule 10.4 applies; otherwise, $G[X \cup \{v\}]$ contains no positive triangles, and Rule 10.5 applies. Finally, if $N_G(v) \cap X = \{x\}$, Rule 10.5 applies for x with clique $G[X \setminus \{x\}]$.

If property B of Lemma 57 holds, then Rule 10.6 applies. If property C of Lemma 57 holds, consider the case when $G - \{v, b\}$ has two connected components. Let Z be the other connected component. If Z is connected to only one of v or b, then property A holds. Otherwise, if $G[X \cup \{x\}]$ contains a positive triangle, where $x \in \{v, b\}$, then by Lemma 56 either Rule 10.1 or Rule 10.2 applies. So we may assume that $G[X \cup \{b, v\}]$ contains no positive triangles, in which case Rule 10.7 applies.

If $G - \{v, b\}$ has at least three connected components, at least two of them, X_1, X_2, form cliques with both v and b and possibly one component Y does not. Assume without loss of generality that Y has an edge to v. Then Rule 10.6 applies for the path $x_1 b x_2$, where $x_1 \in X_1$, $x_2 \in X_2$. □

The following lemma gives structural results on S and $G - S$. Note that from now on, $(G = (V, E), k)$ denotes the original instance of Signed Max Cut ATLB and $(G' = (V', E'), k')$ denotes the instance obtained by applying Rules 10.1-10.7 exhaustively. The set $S \subseteq V$ denotes the set of vertices which are marked by the rules.

Lemma 59. *Given a connected graph G, if we apply Rules 10.1-10.7 exhaustively, either the set S of marked vertices has cardinality at most $3k$, or $k' \leq 0$. In addition, $G - S$ is a forest of cliques that does not contain a positive triangle.*

Proof. Observe that for every reduction rule where some vertices are marked, at most 3 vertices are marked, and the parameter descreases by at least 1. This means that if $k' > 0$, then the reduction rules cannot have marked more than $3k$ vertices.

To show that $G - S$ is a forest of cliques that does not contain a positive triangle, proceed by induction. It is trivially true that the empty graph and the graph with only one vertex are forests

of cliques that do not contain positive triangles. Now suppose that we apply one of the rules, transforming a graph G_1 into a graph G_2; suppose in addition that $G_2 - (S \cap V(G_2))$ is a forest of cliques that does not contain a positive triangle: we claim that $G_1 - (S \cap V(G_1))$ is a forest of cliques that does not contain a positive triangle, too. In the case of Rules 10.1, 10.3, 10.4 and 10.6, $G_1 - (S \cap V(G_1))$ is equal to $G_2 - (S \cap V(G_2))$, therefore the claim is trivially true. For Rule 10.5, note that $G_1 - (S \cap V(G_1))$ is obtained from $G_2 - (S \cap V(G_2))$ by either adding a disjoint clique not containing a positive triangle if $v \in S$, or adding a clique not containing a positive triangle and identifying one of its vertices with v (where v is a cutvertex as in the description of Rule 10.5). Finally, for Rules 10.2 and 10.7, $G_1 - (S \cap V(G_1))$ is obtained from $G_2 - (S \cap V(G_2))$ by adding one disjoint clique not containing a positive triangle. □

Finally, it is possible to prove that Signed Max Cut ATLB is FPT. First we state Max-Cut-with-Weighted-Vertices as in [29].

Max-Cut-with-Weighted-Vertices
Instance: A graph G with weight functions $w_1 : V(G) \to \mathbb{N}_0$ and $w_2 : V(G) \to \mathbb{N}_0$, and an integer $t \in \mathbb{N}$.
Question: Does there exist an assignment $f : V(G) \to \{1, 2\}$ such that $\sum_{xy \in E} \lvert f(x) - f(y) \rvert + \sum_{f(x)=1} w_1(x) + \sum_{f(x)=2} w_2(x) \geq t$?

Theorem 40. Signed Max Cut ATLB *can be solved in time* $O^*(8^k)$.

Proof. Let $(G = (V, E), k)$ be an instance of Signed Max Cut ATLB. Apply Rules 10.1-10.7 exhaustively, producing an instance $(G' = (V', E'), k')$ and a set $S \subseteq V$ of marked vertices. If $k' \leq 0$, (G', k') is a trivial Yes-instance. Since the rules are safe, it follows that (G, k) is a Yes-instance, too.

Otherwise, $k' > 0$. Note that by Lemma 59, $|S| \leq 3k$ and $G - S$ is a forest of cliques, which is a chordal graph without positive triangles. Hence, by Corollary 6, we may assume that $G - S$ does not contain positive edges.

Therefore, to solve Signed Max Cut ATLB on G, we can guess a balanced subgraph of $G[S]$, induced by a partition (V_1, V_2), and then solve Max-Cut-with-Weighted-Vertices for $G - S$. The weight of a vertex $v \in V(G - S)$ is defined in the following way: let $n_i^+(v)$ be the number of positive neighbors of v in V_i and $n_i^-(v)$ be the number of negative neighbors of v in V_i; then $w_1(v) = n_1^+(v) + n_2^-(v)$ and $w_2(v) = n_2^+(v) + n_1^-(v)$.

Since Max-Cut-with-Weighted-Vertices is solvable in polynomial time on a forest of cliques (see Lemma 9 in [29]) and the number of possible partitions of S is bounded by 2^{3k}, this gives an $O^*(8^k)$-algorithm to solve Signed Max Cut ATLB. □

10.4 Kernelization

In this section, we show that Signed Max Cut ATLB admits a kernel with $O(k^3)$ vertices. The proof of Theorem 40 implies the following key result for our kernelization.

Corollary 7. *Let $(G = (V, E), k)$ be an instance of* Signed Max Cut ATLB. *In polynomial time, either we can conclude that (G, k) is a* Yes*-instance or we can find a set S of at most $3k$ vertices for which we may assume that $G - S$ is a forest of cliques without positive edges.*

The kernel is obtained via the application of a new set of reduction rules and using structural results that bound the size of No-instances (G, k). First, we need some additional terminology. For a block C in $G - S$, let $C_{\mathsf{int}} = \{x \in V(C) : N_{G-S}(x) \subseteq V(C)\}$ be the *interior* of C, and let $C_{\mathsf{ext}} = V(C) \setminus C_{\mathsf{int}}$ be the *exterior* of C. If a block C is such that $C_{\mathsf{int}} \cap N_G(S) \neq \emptyset$, C is a *special* block. We say a block C is a *path block* if $|V(C)| = 2 = |C_{\mathsf{ext}}|$. A *path vertex* is a vertex which is contained only in path blocks. A block C in $G - S$ is a *leaf block* if $|C_{\mathsf{ext}}| \leq 1$.

The following reduction rules are two-way reduction rules: they apply to an instance (G, k) and produce an equivalent instance (G', k').

Reduction Rule 10.8. *Let C be a block in $G - S$. If there exists $X \subseteq C_{\mathsf{int}}$ such that $|X| > \frac{|V(C)|+|N_G(X)\cap S|}{2} \geq 1$, $N_G^+(x) \cap S = N_G^+(X) \cap S$ and $N_G^-(x) \cap S = N_G^-(X) \cap S$ for all $x \in X$, then delete two arbitrary vertices $x_1, x_2 \in X$ and set $k' = k$.*

Reduction Rule 10.9. *Let C be a block in $G - S$. If $|V(C)|$ is even and there exists $X \subseteq C_{\mathsf{int}}$ such that $|X| = \frac{|V(C)|}{2}$ and $N_G(X) \cap S = \emptyset$, then delete a vertex $x \in X$ and set $k' = k - 1$.*

Reduction Rule 10.10. *Let C be a block in $G - S$ with vertex set $\{x, y, u\}$, such that $N_G(u) = \{x, y\}$. If the edge xy is a bridge in $G - \{u\}$, delete C, add a new vertex z, positive edges $\{zv : v \in N_{G-u}^+(\{x, y\})\}$, negative edges $\{zv : v \in N_{G-u}^-(\{x, y\})\}$ and set $k' = k$. Otherwise, delete u and the edge xy and set $k' = k - 1$.*

Reduction Rule 10.11. *Let T be a connected component of $G - S$ only adjacent to a vertex $s \in S$. Form a* Max-Cut-with-Weighted2-Vertices *instance on T by defining $w_1(x) = 1$ if $x \in N_G^+(s) \cap T$ ($w_1(x) = 0$ otherwise) and $w_2(y) = 1$ if $y \in N_G^-(s) \cap T$ ($w_2(y) = 0$ otherwise). Let $\beta(G[V(T) \cup \{s\}]) = \mathsf{pt}(G[V(T) \cup \{s\}]) + \frac{p}{4}$. Then delete T and set $k' = k - p$.*

Note that the value of p in Rule 10.11 can be found in polynomial time by solving Max-Cut-with-Weighted-Vertices on T.

A two-way reduction rule is *valid* if it transforms Yes-instances into Yes-instances and No-instances into No-instances. Theorem 41 shows that Rules 10.8-10.11 are valid. To prove Theorem 41, we need the following two lemmas.

Lemma 60. *Let C be a block in $G - S$. If there exists $X \subseteq C_{\mathsf{int}}$ such that $|X| \geq \frac{|V(C)|}{2}$, then there exists a (V_1, V_2)-balanced subgraph H of G with $\beta(G)$ edges such that at least one of the following inequalities holds:*

- $|V_2 \cap V(C)| \leq |V_1 \cap V(C)| \leq |N_G(X) \cap S| + |V_2 \cap V(C)|$;
- $|V_2 \cap V(C)| \leq |V_1 \cap V(C)| \leq |V_2 \cap V(C)| + 1$.

Proof. We may assume that $|V_1 \cap V(C)| \geq |V_2 \cap V(C)|$. Note that if $|V_1 \cap V(C)| > |V_2 \cap V(C)|$, then $X \cap V_1 \neq \emptyset$ (because $|X| \geq \frac{|V(C)|}{2}$).

First, if $N_G(X) \cap S = \emptyset$ and $|V_1 \cap V(C)| \geq |V_2 \cap V(C)| + 2$, then, for any $x \in X \cap V_1$, the subgraph induced by the partition $(V_1 \setminus \{x\}, V_2 \cup \{x\})$ has more edges than the subgraph induced by (V_1, V_2), which is a contradiction.

Now, suppose that $N_G(X) \cap S \neq \emptyset$ and suppose also that $|V_1 \cap V(C)| - |V_2 \cap V(C)|$ is minimal. If $|V_1 \cap V(C)| \leq |V_2 \cap V(C)| + 1$ we are done, so suppose $|V_1 \cap V(C)| \geq |V_2 \cap V(C)| + 2$. Consider the partition $V_1' = V_1 \setminus \{x\}$, $V_2' = V_2 \cup \{x\}$, where $x \in V_1 \cap X$, and the balanced subgraph H' induced by this partition. Then $|E(H')| \geq |E(H)| + |E(V_1 \setminus \{x\}, x)| - |E(V_2, x)| \geq |E(H)| + (|V_1 \cap V(C)| - 1 - |N_G(X) \cap S| - |V_2 \cap V(C)|)$. Since $|V_1' \cap V(C)| - |V_2' \cap V(C)| < |V_1 \cap V(C)| - |V_2 \cap V(C)|$, it holds that $|E(H')| \leq |E(H)| - 1$. Therefore, $|V_1 \cap V(C)| \leq |N_G(X) \cap S| + |V_2 \cap V(C)|$. □

Lemma 61. *Let C be a block in $G - S$. If there exists $X \subseteq C_{int}$ such that $|X| > \frac{|V(C)| + |N_G(X) \cap S|}{2}$, $N_G^+(x) \cap S = N_G^+(X) \cap S$ and $N_G^-(x) \cap S = N_G^-(X) \cap S$ for all $x \in X$, then, for any $x_1, x_2 \in X$, there exists a (V_1, V_2)-balanced subgraph H of G with $\beta(G)$ edges such that $x_1 \in V_1$ and $x_2 \in V_2$.*

Proof. First, we claim that there exist vertices $x_1, x_2 \in X$ for which the result holds. Let H be a (V_1, V_2)-balanced subgraph of G with $\beta(G)$ edges as given by Lemma 60.

Suppose $N_G(X) \cap S = \emptyset$. Then, by Lemma 60 it holds that $|V_2 \cap V(C)| \leq |V_1 \cap V(C)| \leq |V_2 \cap V(C)| + 1$; in addition, $|X| > \frac{|V(C)|}{2}$. Hence, either we can find x_1 and x_2 as required, or $X = V_1 \cap V(C)$ and $|V_1 \cap V(C)| = |V_2 \cap V(C)| + 1$. In the second case, pick a vertex $x \in V_1$ and form the partition $V_1' = V_1 \setminus \{x\}$ and $V_2' = V_2 \cup \{x\}$. Consider the balanced subgraph H' induced by this partition. Observe that $|E(H')| = |E(H)| - |E(x, V_2)| + |E(x, V_1 \setminus \{x\})| = |E(H)| - |V_2 \cap V(C)| + |V_1 \cap V(C)| - 1 = |E(H)|$, so H' is a maximum balanced subgraph for which we can find x_1 and x_2 as required.

Now, suppose $N_G(X) \cap S \neq \emptyset$. Then by Lemma 60 it holds that $|V_2 \cap V(C)| \leq |V_1 \cap V(C)| \leq |N_G(X) \cap S| + |V_2 \cap V(C)|$. For the sake of contradiction, suppose $X \subseteq V_1 \cap V(C)$ or $X \subseteq V_2 \cap V(C)$: in both cases, this means that $|X| \leq |V_1 \cap V(C)|$. Note that $|V(C)| = |V_1 \cap V(C)| + |V_2 \cap V(C)| = 2|V_2 \cap V(C)| + t$, where $t \leq |N_G(X) \cap S|$. Hence, $|V_1 \cap V(C)| \geq |X| > \frac{|V(C)| + |N_G(X) \cap S|}{2} = |V_2 \cap V(C)| + \frac{t}{2} + \frac{|N_G(X) \cap S|}{2} \geq |V_2 \cap V(C)| + t = |V_1 \cap V(C)|$, which is a contradiction.

To conclude the proof, notice that for a (V_1, V_2)-balanced subgraph H of G with $\beta(G)$ edges and vertices $x_1, x_2 \in X$ such that $x_1 \in V_1$ and $x_2 \in V_2$, we have $|E(H)| = |E(H')|$, where H' is a balanced subgraph induced by $V_1' = V_1 \setminus \{x_1\} \cup \{x_2\}$ and $V_2' = V_2 \setminus \{x_2\} \cup \{x_1\}$: this is true because $N_G^+(x_1) \cap S = N_G^+(x_2) \cap S$ and $N_G^-(x_1) \cap S = N_G^-(x_2) \cap S$. □

Theorem 41. *Rules 10.8-10.11 are valid.*

Proof. **Rule 10.8:** Let C, X be as in the description of Rule 10.8. Let $x_1, x_2 \in X$. By Lemma 61, there exists a (V_1, V_2)-balanced subgraph H of G with $\beta(G)$ edges such that $x_1 \in V_1$ and

$x_2 \in V_2$. Now, let $G' = G - \{x_1, x_2\}$ and $H' = H - \{x_1, x_2\}$. Since $N_G^+(x_1) \cap S = N_G^+(x_2) \cap S$ and $N_G^-(x_1) \cap S = N_G^-(x_2) \cap S$, it holds that $|E(H)| = |E(H')| + \frac{|E(G,\{x_1,x_2\})|}{2} + 1$, and so $\beta(G') + \frac{|E(G,\{x_1,x_2\})|}{2} + 1 \geq \beta(G)$. Conversely, by Lemma 54, $\beta(G) \geq \beta(G') + \frac{|E(G,\{x_1,x_2\})|}{2} + 1$. Finally, observe that $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{|E(G,\{x_1,x_2\})|}{2} + 1$, which implies that $\beta(G) - \mathsf{pt}(G) = \beta(G') - \mathsf{pt}(G')$. Hence, G admits a balanced subgraph of size $\mathsf{pt}(G) + \frac{k}{4}$ if and only if G' admits a balanced subgraph of size $\mathsf{pt}(G') + \frac{k}{4}$.

Rule 10.9: Let C, X and $x \in X$ be as in the description of Rule 10.9. By Lemma 60, there exists a (V_1, V_2)-balanced subgraph H of G with $\beta(G)$ edges, such that $|V_1 \cap V(C)| = |V_2 \cap V(C)|$. Consider the graph $G' = G - \{x\}$ formed by the application of the rule and the balanced subgraph $H' = H - \{x\}$. Then $|E(H)| = |E(H')| + \frac{|V(C)|}{2}$, and thus $\beta(G') \geq \beta(G) - \frac{|V(C)|}{2}$. Conversely, by Lemma 54, $\beta(G) \geq \beta(G') + \frac{|V(C)|}{2}$. However, $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{|V(C)|}{2} - \frac{1}{4}$. Hence, $\beta(G) - \mathsf{pt}(G) = \beta(G') - \mathsf{pt}(G') + \frac{1}{4}$. Therefore, G admits a balanced subgraph of size $\mathsf{pt}(G) + \frac{k}{4}$ if and only if G' admits a balanced subgraph of size $\mathsf{pt}(G') + \frac{k-1}{4}$.

Rule 10.10: Let C and $\{x, y, u\}$ be as in the description of Rule 10.10. Firstly consider the case when xy is a bridge in $G - \{u\}$. For any maximal balanced subgraph H of G, without loss of generality one may assume that $xu, yu \in E(H)$ and $xy \notin E(H)$. Suppose H is induced by a partition (V_1, V_2) and $x, y \in V_1$. Form a balanced subgraph of G' from $H - \{x, y, u\}$ by placing z in V_1. Therefore, $\beta(G) = \beta(G') + 2$. Since $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{3}{2} + \frac{2}{4} = \mathsf{pt}(G') + 2$, it follows that $\beta(G) = \mathsf{pt}(G) + \frac{k}{4}$ if and only if $\beta(G') = \mathsf{pt}(G') + \frac{k}{4}$.

Now consider the case when xy is not a bridge in $G - \{u\}$. Then the graph G' formed by deleting the vertex u and the edge xy is connected. Furthermore, regardless of whether x and y are in the same partition that induces a balanced subgraph H' of G', H' can be extended to a balanced subgraph H of G such that $|E(H)| = |E(H')| + 2$. This means that, as before, $\beta(G) = \beta(G') + 2$. But in this case $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{7}{4}$ and thus $\beta(G) = \mathsf{pt}(G) + \frac{k}{4}$ if and only if $\beta(G') = \mathsf{pt}(G') + \frac{k-1}{4}$.

Rule 10.11: Let T and $s \in S$ be as in the description of Rule 10.11. Since $\beta(G[V(T) \cup \{s\}]) = \mathsf{pt}(G[V(T) \cup \{s\}]) + \frac{p}{4}$, by Lemma 55, $\beta(G) = \beta(G - T) + \mathsf{pt}(G[V(T) \cup \{s\}]) + \frac{p}{4}$. Also, by Lemma 55, $\mathsf{pt}(G) = \mathsf{pt}(G - T) + \mathsf{pt}(G[V(T) \cup \{s\}])$. Hence $\beta(G) - \mathsf{pt}(G) = \beta(G - T) - \mathsf{pt}(G - T) + \frac{p}{4}$, which implies that G admits a balanced subgraph of size $\mathsf{pt}(G) + \frac{k}{4}$ if and only if $G - T$ admits a balanced subgraph of size $\mathsf{pt}(G - T) + \frac{k-p}{4}$. □

To show the existence of a kernel with $O(k^3)$ vertices, it is enough to give a bound on the number of non-path blocks, the number of vertices in these blocks and the number of path vertices. This is done by Corollaries 9 and 10 and Lemma 67.

While Lemma 67 applies to any graph reduced by Rule 10.8, the proofs of Corollaries 9 and 10 rely on Lemma 66, which gives a general structural result on forest of cliques with a bounded number of special blocks and bounded path length. Corollary 8 and Lemma 64 provide sufficient conditions for a reduced instance to be a YES-instance, thus producing a bound on the number of special blocks and the path length of NO-instances. Lastly, Theorem 42 puts the results together to show the existence of the kernel.

Henceforth, we assume that the instance (G, k) is such that G is reduced by Rules 10.8-10.11, $G - S$ is a forest of cliques which does not contain a positive edge and $|S| \leq 3k$.

Lemma 62. *Let T be a connected component of $G - S$. Then for every leaf block C of T, $N_G(C_{int}) \cap S \neq \emptyset$. Furthermore, if $|N_G(S) \cap V(T)| = 1$, then T consists of a single vertex.*

Proof. We start by proving the first claim. Note that if $T = C$ consists of a single vertex, then $N_G(C_{\mathsf{int}}) \cap S \neq \emptyset$ since G is connected. So assume that C has at least two vertices. Suppose that $N_G(C_{\mathsf{int}}) \cap S = \emptyset$ and let $X = C_{\mathsf{int}}$. Then if $|C_{\mathsf{int}}| > |C_{\mathsf{ext}}|$, Rule 10.8 applies. If $|C_{\mathsf{int}}| = |C_{\mathsf{ext}}|$ then Rule 10.9 applies. Otherwise, $|C_{\mathsf{int}}| < |C_{\mathsf{ext}}|$ and since $|C_{\mathsf{ext}}| \leq 1$ (as C is a leaf block), C has only one vertex, which contradicts our assumption above. For the second claim, first note that since $|N_G(S) \cap V(T)| = 1$, Q has one leaf block and so T consists of a single block. Let $N_G(S) \cap V(T) = \{v\}$ and $X = V(T) - \{v\}$. If $|X| > 1$, Rule 10.8 applies. If $|X| = 1$, Rule 10.9 applies. Hence $V(T) = \{v\}$. □

Let $\mathcal{B}$ be the set of non-path blocks.

Lemma 63. *If there exists a vertex $s \in S$ such that $\sum_{C \in \mathcal{B}} |N_G(C_{int}) \cap \{s\}| \geq 2(|S| - 1 + k)$, then (G, k) is a* YES*-instance.*

Proof. Form $T \subseteq N_G(s)$ by picking a vertex from each block C for which $|N_G(C_{\mathsf{int}}) \cap \{s\}| = 1$: if there exists a vertex $x \in C_{\mathsf{int}}$ such that $N_G(x) \cap S = \{s\}$, pick this, otherwise pick $x \in C_{\mathsf{int}}$ arbitrarily. Let $U = T \cup \{s\}$ and $W = V \setminus U$.

Observe that $G[U]$ is balanced by Theorem 36 as $G[U]$ is a tree. Thus $\beta(G[U]) = |T| = \frac{|T|}{2} + \frac{|T|}{4} + \frac{|T|}{4} = \mathsf{pt}(G[U]) + \frac{|T|}{4}$.

Consider a connected component Q of $G - S$. By Rule 10.11, $|N_G(Q) \cap S| \geq 2$ and by Lemma 62, if $|N_G(S) \cap V(Q)| = 1$ then Q consists of a single vertex. Otherwise, either $(N_G(S) \setminus N_G(s)) \cap V(Q) \neq \emptyset$, or Q has at least two vertices in T. Moreover, note that the removal of interior vertices does not disconnect the component itself. Hence $G[W]$ has at most $(|S| - 1) + \frac{|T|}{2}$ connected components. Applying Lemma 54, $\beta(G) \geq \mathsf{pt}(G) + \frac{|T|}{4} - \frac{(|S|-1)+\frac{|T|}{2}}{4} = \mathsf{pt}(G) + \frac{|T|}{8} - \frac{|S|-1}{4}$. Hence if $|T| \geq 2(|S| - 1 + k)$, then (G, k) is a YES-instance. □

Corollary 8. *If $\sum_{C \in \mathcal{B}} |N_G(C_{int}) \cap S| \geq |S|(2|S| - 3 + 2k) + 1$, the instance is a* YES*-instance. Otherwise, $\sum_{C \in \mathcal{B}} |N_G(C_{int}) \cap S| \leq 3k(8k - 3)$.*

Proof. If $\sum_{C \in \mathcal{B}} |N_G(C_{\mathsf{int}}) \cap S| \geq |S|(2|S| - 3 + 2k) + 1$, then for some $s \in S$ we have $\sum_{C \in \mathcal{B}} |N_G(C_{\mathsf{int}}) \cap \{s\}| \geq 2|S| - 3 + 2k + 1/|S|$ and, since the sum is integral, $\sum_{C \in \mathcal{B}} |N_G(C_{\mathsf{int}}) \cap \{s\}| \geq 2(|S| - 1 + k)$. Thus, (G, k), by Lemma 63, is a YES-instance. The second inequality of the corollary follows from the fact that $|S| \leq 3k$. □

Lemma 64. *If in $G - S$ there exist vertices $U = \{u_1, u_2, \ldots, u_p\}$ such that $N_{G-S}(u_i) = \{u_{i-1}, u_{i+1}\}$ for $2 \leq i \leq p - 1$, and $p \geq |S| + k + 1$, then (G, k) is* YES*-instance. Otherwise, $p \leq 4k$.*

Proof. Observe that $G[U]$ is balanced by Theorem 36. Thus $\beta(G[U]) = p - 1 = \mathsf{pt}(G[U]) + \frac{p-1}{4}$. Let $W = V \setminus U$ and observe that $G[W]$ has at most $|S|$ components, since, by Lemma 62, every vertex in $G - U$ has a path to a vertex in S. Applying Lemma 54, $\beta(G) \geq \mathsf{pt}(G) + \frac{p-1}{4} - \frac{|S|}{4}$. Hence if $p - 1 - |S| \geq k$, (G, k) is a YES-instance. □

Lemma 65. *A block C in $G - S$ such that $|C_{ext}| = 2$ is either special or it is a path block.*

Proof. Suppose C is not special. If $|V(C)| \geq 5$, then Reduction Rule 10.8 would apply. If $|V(C)| = 4$, then Reduction Rule 10.9 would apply. If $|V(C)| = 3$, then Reduction Rule 10.10 would apply. Hence $|V(C)| = 2$ and it is a path block. □

In $G - S$, a *pure path* is a path consisting exclusively of path vertices. Note that every path vertex belongs to a unique pure path.

Lemma 66. *Suppose $G - S$ has at most l special blocks and the number of vertices in each pure path is bounded by p. Then $G - S$ contains at most $2l$ non-path blocks and $2pl$ path vertices.*

Proof. It suffices to prove that if every connected component T of $G - S$ has at most l_T special blocks, then T contains at most $2l_T$ non-path blocks and $2pl_T$ path vertices. So, we may assume that $T = G - S$ is connected. Pick an arbitrary non-path block C_R as the 'root' node. Define the distance $d(C_R, C)$ as the number of non-path blocks different from C_R visited in a path from a vertex in C_R to a vertex in C. For every non-path block C in T, the *parent* block C' is the unique non-path block such that C' contains an edge of any path from C_R to C and $d(C_R, C) - d(C_R, C') = 1$. In addition, C_R is the parent of every block C such that $d(C_R, C) = 1$.

Consider the tree F that contains a vertex for every non-path block of T and such that there is an edge between two vertices if and only if one of the corresponding blocks is the parent of the other. Observe that given a vertex $v \in F$ which corresponds to a block C of T, it holds that $d_F(v) \geq |C_{\mathsf{ext}}|$. In addition, by Lemma 62, every leaf in F corresponds to a special block.

Now, we know that in a tree the number of vertices of degree greater or equal to three is bounded by the number of leaves. Moreover, by Lemma 65, if a block C is such that $|C_{\mathsf{ext}}| = 2$, then it is either special or a path block. Thus, the number of non-path blocks is bounded by $2l$.

Furthermore, note that the number of pure paths in T is bounded by the number of edges in F, which is bounded by $2l - 1$. Since every pure path contains at most p path vertices, the number of path vertices is bounded by $(2l - 1)p < 2pl$. □

Corollary 9. *$G - S$ contains at most $6k(8k - 3)$ non-path blocks and $24k^2(8k - 3)$ path vertices.*

Proof. By Corollary 8, $G - S$ contains at most $3k(8k - 3)$ special blocks and by Lemma 64, the length of every pure path is bounded by $4k$. Thus, Lemma 66 implies that $G - S$ contains at most $6k(8k - 3)$ non-path blocks and $24k^2(8k - 3)$ path vertices. □

Corollary 10. *$G - S$ contains at most $12k(8k - 3)$ vertices in the exteriors of non-path blocks.*

Proof. For any component T of $G - S$, consider the tree F defined in the proof of Lemma 66. For any block C of T and any vertex v in C_{ext}, v corresponds to an edge of F. Furthermore, for any edge of F there are at most two exterior vertices in T that correspond to it. Therefore, $|\cup_{C\in\mathcal{B}} C_{\mathsf{ext}}| \leq 2|\mathcal{B}| \leq 12k(8k-3)$. □

Lemma 67. *For a block C, if $|V(C)| \geq 2|C_{ext}| + |N_G(C_{int}) \cap S|(2|S| + 2k + 1)$, then (G, k) is a* Yes*-instance. Otherwise, $|V(C)| \leq 2|C_{ext}| + |N_G(C_{int}) \cap S|(8k + 1)$.*

Proof. Consider a fixed $s \in N_G(C_{\mathsf{int}}) \cap S$. We will show that we may assume that either $|N_G^+(s) \cap C_{\mathsf{int}}| \leq \frac{k+|S|}{2}$ or $|N_G^+(s) \cap C_{\mathsf{int}}| \geq |C_{\mathsf{int}}| - \frac{k+|S|}{2}$, because otherwise (G, k) is a Yes-instance.

Indeed, suppose $\lceil \frac{k+|S|}{2} \rceil \leq |N_G^+(s) \cap C_{\mathsf{int}}| \leq |C_{\mathsf{int}}| - \lceil \frac{k+|S|}{2} \rceil$. Let $U_1 \subseteq N_G^+(s) \cap C_{\mathsf{int}}$, $|U_1| = \lceil \frac{k+|S|}{2} \rceil$, and let $U_2 \subseteq C_{\mathsf{int}} \setminus N_G^+(s)$, $|U_2| = \lceil \frac{k+|S|}{2} \rceil$. Let $U = U_1 \cup U_2 \cup \{s\}$ and consider the subgraph H of $G[U]$ induced by the edges $E(U_1, U_2) \cup E(s, (U_1 \cap N_G^+(s))) \cup E(s, (U_2 \cap N_G^-(s)))$. Observe that H is $(U_1 \cup \{s\}, U_2)$-balanced and so $\beta(G[U]) \geq |U_1|^2 + |U_1 \cap N_G^+(s)| + |U_2 \cap N_G^-(s)|$. Furthermore, $\mathsf{pt}(G[U]) = |U_1|^2 + \frac{|U_1 \cap N_G^+(s)|}{2} + \frac{|U_2 \cap N_G^-(s)|}{2}$, and hence $\beta(G[U]) \geq \mathsf{pt}(G[U]) + \frac{|U_1 \cap N_G^+(s)| + |U_2 \cap N_G^-(s)|}{2} \geq \mathsf{pt}(G[U]) + \frac{k+|S|}{4}$.

Now consider $W = V \setminus U$. Any connected component of $G - S$ is connected to two vertices in S, hence $G[W]$ has at most $|S| - 1$ components adjacent to vertices in $S \setminus \{s\}$ and one component corresponding to the block C. Applying Lemma 54, $\beta(G) \geq \mathsf{pt}(G) + \frac{(k+|S|)-|S|}{4}$, which means that (G, k) is a Yes-instance.

Similarly, we can show that we may assume that either $|N_G^-(s) \cap C_{\mathsf{int}}| \leq \frac{k+|S|}{2}$ or $|N_G^-(s) \cap C_{\mathsf{int}}| \geq |C_{\mathsf{int}}| - \frac{k+|S|}{2}$, because otherwise (G, k) is a Yes-instance.

Let $S_1^+ = \{s \in S : 0 < |N_G^+(s) \cap C_{\mathsf{int}}| \leq \frac{k+|S|}{2}\}$, $S_2^+ = (N_G^+(C_{\mathsf{int}}) \cap S) \setminus S_1^+$ and $X^+ = \{v \in C_{\mathsf{int}} \setminus N_G^+(S_1^+) : v \in N_G^+(s), \forall s \in S_2^+\}$. Observe that for all $s \in S_2^+$, $|N_G^+(s) \cap C_{\mathsf{int}}| \geq |C_{\mathsf{int}}| - \frac{k+|S|}{2}$, which means that $|X^+| \geq |C_{\mathsf{int}} \setminus N_G^+(S_1^+)| - |S_2^+| \frac{k+|S|}{2}$. In addition, $|N_G^+(S_1^+) \cap C_{\mathsf{int}}| \leq |S_1^+| \frac{k+|S|}{2}$, hence $|C_{\mathsf{int}} \setminus N_G^+(S_1^+)| \geq |C_{\mathsf{int}}| - |S_1^+| \frac{k+|S|}{2}$. Therefore, $|X^+| \geq |C_{\mathsf{int}}| - (|S_1^+| + |S_2^+|) \frac{k+|S|}{2} = |C_{\mathsf{int}}| - |N_G^+(C_{\mathsf{int}}) \cap S| \frac{k+|S|}{2} \geq |C_{\mathsf{int}}| - |N_G(C_{\mathsf{int}}) \cap S| \frac{k+|S|}{2}$.

With similar definitions and the same argument we obtain $|X^-| \geq |C_{\mathsf{int}}| - |N_G(C_{\mathsf{int}}) \cap S| \frac{k+|S|}{2}$. Now let $X = X^+ \cap X^-$ and observe that and $|X| \geq |C_{\mathsf{int}}| - |N_G(C_{\mathsf{int}}) \cap S|(k + |S|)$.

However, by Rule 10.8, $|X| \leq \frac{|V(C)|+|N_G(C_{\mathsf{int}}) \cap S|}{2}$. So, $|C_{\mathsf{int}}| \leq |N_G(C_{\mathsf{int}}) \cap S|(|S| + k + \frac{1}{2}) + \frac{|V(C)|}{2}$, and so $|V(C)| \leq 2|C_{\mathsf{ext}}| + |N_G(C_{\mathsf{int}}) \cap S|(2|S| + 2k + 1)$ as claimed. □

Theorem 42. Signed Max Cut ATLB *has a kernel with $O(k^3)$ vertices.*

Proof. Let $(G = (V, E), k)$ be an instance of Signed Max Cut ATLB. As in Theorem 40, apply Rules 10.1-10.7 exhaustively: either the instance is a Yes-instance, or there exists $S \subseteq V$ such that $|S| \leq 3k$ and $G - S$ is a forest of cliques which does not contain a positive edge.

Now, apply Rules 10.8–10.11 exhaustively to (G, k) to obtain a new instance (G', k'). If $k' \leq 0$, then (G, k) is a Yes-instance since Rules 10.8–10.11 are valid. Now let $G = G', k = k'$. Check whether (G, k) is a Yes-instance due to Corollary 8, Lemma 64 or Lemma 67. If this is not the case, by Corollary 9, $G - S$ contains at most $6k(8k - 3)$ non-path blocks and $24k^2(8k - 3)$ path

vertices. Hence, by Lemma 67, $|V(G)|$ is at most

$$|S| + 24k^2(8k-3) + \sum_{C \in \mathcal{B}} |V(C)| \leq O(k^3) + 2 \sum_{C \in \mathcal{B}} |C_{\mathsf{ext}}| + (8k+1) \sum_{C \in \mathcal{B}} |N_G(C_{\mathsf{int}}) \cap S|$$

Now, applying Corollary 8 and Corollary 10, we obtain:

$$|V(G)| \leq O(k^3) + 48k(8k-3) + 3k(8k-3)(8k+1) = O(k^3).$$

□

It is not hard to verify that none of our reduction rule increase the number of positive edges. Thus, considering an input G of Max Cut ATLB as an input of Signed Max Cut ATLB by assigning minus to each edge of G, we have the following:

Corollary 11. Max Cut ATLB *has a kernel with* $O(k^3)$ *vertices.*

Chapter 11

Discussion and Future Work

11.1 MaxLin2

In Chapter 4 we studied the problem MaxLin2 parameterized above average. Whilst the kernels obtained in Theorems 6 and 8 bounded the number of variables by a polynomial, they did not bound the number of equations, and are thus not polynomial *size* kernels. The question of the existence of polynomial-size kernels for MaxLin2-AA[k] and Max-r-Lin2-AA[k, r] remains an interesting open question.

One could also look into improving the kernel - perhaps by finding a structural characterization of irreducible system with excess bounded by k. Indeed, understanding the structural properties of irreducible systems would be of interest independently of any gain in the size of a kernel.

In a graph $G = (V, E)$, a bisection (X, Y) is a partition of V into sets X and Y such that $|X| \leq |Y| \leq |X| + 1$. The size of (X, Y) is the number of edges between X and Y. In Max Bisection, we are given a graph G with $n \geq 2$ vertices and m edges and asked to find a bisection of maximum size. It is not hard to see that $\lceil m/2 \rceil$ is a tight lower bound on the maximum size of a bisection of G. Gutin and Yeo [53] proved that Max Bisection parameterized above $\lceil m/2 \rceil$ has a kernel with $O(k^2)$ vertices and $O(k^3)$ edges. Gutin and Yeo [53] also showed that $\lceil \frac{nm}{2(n-1)} \rceil$ is another tight lower bound on the maximum size of a bisection of G. Clearly, $\lceil \frac{nm}{2(n-1)} \rceil \geq \lceil m/2 \rceil$. Gutin and Yeo [53] left it as an open problem to determine the complexity of Max Bisection parameterized above $\lceil \frac{nm}{2(n-1)} \rceil$.

In Chapter 5 we focused on MaxLin2 parameterized below the total weight of the system. Our study was thorough, showing that Max-(= 3,= 3)-Lin2-B[m] is W[1]-hard, but Max-($\leq$ 2,*)-Lin2-BW is fixed-parameter tractable and Max-(*,$\leq$ 2)-Lin2 is polynomial time solvable. This gives a boundary between parameterized intractability and tractability for MaxLin2-BW.

11.2 MaxSAT

In Chapter 6 we considered the problem MaxSAT parameterized above average. Our results leave a small set of open questions. There is a gap between the inequalities of Theorem 16 and 18(i), it would be interesting to close this gap. We also would like to know whether Max-$r(n)$Sat-AA is fixed-parameter tractable if $r(n) \leq \log\log n - \log\log\log n$ (recall currently we know Max-$r(n)$Sat-AA is fixed-parameter tractable if $r(n) \leq \log\log n - \log\log\log n - \varphi(n)$).

Apart from MaxLin-AA and MaxSat-AA mentioned above, there are some other constraint satisfaction problems parameterized above a tight lower bound whose complexity has been established in the last few years. One example is r-Linear-Ordering-AA for $r = 2$ and 3. Let $r \geq 2$ be a fixed integer. In r-Linear-Ordering, given a positive integer n and a multiset $\mathcal{C}$ of r-tuples of distinct elements from $[n]$, we wish to find a permutation $L : [n] \to [n]$ which maximizes that the number of *satisfied* r-tuples, i.e., r-tuples $(i_1, i_2, \ldots, i_r)$ such that $L(i_1) < L(i_2) < \cdots < L(i_r)$. Let m stand for the number of r-tuples in $\mathcal{C}$.

Let $\tau : [n] \to [n]$ be a random permutation (chosen uniformly from the set of all permutations). Observe that the probability that an r-tuple is satisfied by τ is $1/r!$. Thus, by linearity of expectation, the expected number of r-tuples satisfied by τ is $m/r!$. Using the conditional expectation derandomization method [5], it is not difficult to obtain a polynomial time algorithm for finding a permutation L which satisfies at least $m/r!$ r-tuples. Thus, we can easily obtain an $1/r!$-approximation algorithm for r-Linear Ordering. It is remarkable that for any positive ϵ there exists no polynomial $(1/r! + \epsilon)$-approximation algorithm provided the Unique Games Conjecture (UGC) of Khot [69] holds. This result was proved by Guruswami *et al.* [57] for $r = 2$, Charikar *et al.* [16] for $r = 3$ and, finally, by Guruswami *et al.* [56] for any r.

Observe that every permutation L satisfies exactly one r-tuple in the set $\{(i_1, i_2, \ldots, i_r) : \{i_1, i_2, \ldots, i_r\} = [r]\}$ and, thus, $m/r!$ is a tight lower bound on the maximum number of r-tuples that can be satisfied by a permutation L. It is natural to ask what is the parameterized complexity of the following problem r-Linear-Ordering-AA: decide whether there is a permutation L which satisfies at least $m/r! + k$ r-tuples, where $k \geq 0$ is the parameter. Gutin *et al.* [49] and [47] proved that r-Linear-Ordering-AA is fixed-parameter tractable for $r = 2$ and $r = 3$, respectively. The complexity of r-Linear-Ordering-AA for $r \geq 4$ remains an open problem [47]. Note that if r-Linear-Ordering-AA is fixed-parameter tractable for some r, then all permutation constraint satisfaction problems of arity r parameterized above average are fixed-parameter tractable too (see Gutin *et al.* [47] for the definition of a permutation constraint satisfaction problem of arity r and a proof of the above-mentioned fact).

In Chapter 7 we showed that for any CNF formula F, it is fixed-parameter tractable to decide if F has a satisfiable subformula containing α clauses, where $\alpha - \nu(F)$ is the parameter. Our result implies fixed-parameter tractability for the problem of deciding satisfiability of F when F is variable-matched and $\delta(F) \leq k$, where k is the parameter. In addition, we show that the problem does not have a polynomial-size kernel unless coNP $\subseteq$ NP/poly.

Clearly, parameterizations of MaxSat above $m/2$ and $\nu(F)$ are "stronger" than the standard parameterization (i.e., when the parameter is the size of the solution). Whilst the two non-standard parameterizations have smaller parameter than the standard one, they are incomparable to each other as for some formulas F, $m/2 < \nu(F)$ (e.g., for variable-matched formulas with $m < 2n$) and for some formulas F, $m/2 > \nu(F)$ (e.g., when $m > 2n$). Recall that Mahajan and Raman [85] proved that MaxSat parameterized above $m/2$ is fixed-parameter tractable. This result and our main result imply that MaxSat parameterized above $\max\{m/2, \nu(F)\}$ is fixed-parameter tractable: if $m/2 > \nu(F)$ then apply the algorithm of [85], otherwise apply our algorithm.

If every clause of a formula with m clauses contains exactly two literals then it is well known that we can satisfy at least $3m/4$ clauses. From this, and by applying Reduction Rules 7.1 and 7.2, we can get a linear kernel for this version of the MaxSat-A$\nu(F)$ problem. It would be nice to see whether a linear or a polynomial-size kernel exists for the MaxSat-A$\nu(F)$ problem if every clause has exactly r literals.

In Chapter 8 we considered Unit-Conflict Free Max-SAT. A CNF formula is unit-conflict free if and only if it is 2-satisfiable. A CNF formula I is *t-satisfiable* if any subset of t clauses of I can be satisfied simultaneously. Let r_t be the largest real such that in any t-satisfiable CNF formula at least r_t-th fraction of its clauses can be satisfied simultaneously. Note that $r_1 = 1/2$ and $r_2 = (\sqrt{5}-1)/2$. Lieberherr and Specker [80] and, later, Yannakakis [109] proved that $r_3 \geq 2/3$. Käppeli and Scheder [68] proved that $r_3 \leq 2/3$ and, thus, $r_3 = 2/3$. Král [76] established the value of r_4: $r_4 = 3/(5 + [(3\sqrt{69} - 11)/2]^{1/3} - [3\sqrt{69} + 11)/2]^{1/3}) \approx 0.6992$.

For general t, Huang and Lieberherr [63] showed that $\lim_{t\to\infty} r_t \leq 3/4$ and Trevisan [105] proved that $\lim_{t\to\infty} r_t = 3/4$ (a different proof of this result is later given by Král [76]).

For a 3-satisfiable CNF formula $F = (V, C)$, Gutin et. al. [46] considered the question of deciding whether $\text{sat}(F) \geq 2|C|/3 + k$, where k is the parameter. Using the probabilistic method, they showed the problem has a kernel with a linear number of variables.

Similar question for any fixed $t > 3$ remains open.

11.3 Problems above Poljak-Turzík

In Chapter 9 we showed a kernel with $O(k^2)$ vertices and $O(k^2)$ arcs for the Acyclic Subgraph above Poljak-Turzík Bound (ASAPT) problem. In Chapter 10 we showed that Signed Max Cut ATLB has a kernel with $O(k^3)$ vertices.

Both of these problems are in the family of problems parameterized above the Poljak-Turzík bound. Independently, Mnich *et al.* [89] have combined modified approaches of [29] and [96] to prove that a number of graph problems parameterized above tight lower bounds are fixed-parameter tractable. In particular, they proved that ASAPT is fixed-parameter tractable. However, [89] did not obtain any results on polynomial kernels.

Our algorithms for Acyclic Subgraph above Poljak-Turzík Bound (ASAPT) (theorem 34) has runtime $2^{O(k \log k)} n^{O(1)}$. It would be interesting to design an algorithm of runtime

$2^{O(k)}n^{O(1)}$ or to prove that such an algorithm does not exist, subject to a certain complexity hypothesis, as in [82]. It would also be interesting to see whether ASAPT admits a kernel with $O(k)$ vertices.

In Chapter 10 the input of Signed Max Cut ATLB is a signed graph without parallel edges. However, in some applications (cf. [44, 55]), signed graphs may have parallel edges of opposite signs. We may easily extend inputs of Signed Max Cut ATLB to such graphs. Indeed, if G is such a graph we may remove all pairs of parallel edges from G and obtain an equivalent instance of Signed Max Cut ATLB. In fact, the Poljak-Turzík bound can be extended to edge-weighted graphs [96]. Let G be a connected signed graph in which each edge e is assigned a positive weight $w(e)$. The weight $w(Q)$ of an edge-weighted graph Q is the sum of weights of its edges. Then G contains a balanced subgraph with weight at least $w(G)/2 + w(T)/4$, where T is a spanning tree of G of minimum weight [96]. It would be interesting to establish parameterized complexities of Max Cut ATLB and Signed Max Cut ATLB extended to edge-weighted graphs using the Poljak-Turzík bound above.

Bibliography

[1] F.N. Abu-Khzam and H. Fernau, Kernels: Annotated, proper and induced. *Proc. IWPEC 2006*, Lect. Notes Comput. Sci. 4169:264–275, 2006.

[2] R. Aharoni and N. Linial, Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Combin. Th. Ser. A*, 43: 196–204, 1986.

[3] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica* 61(3): 638–655 (2011).
A preliminary version was published in *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pp. 511–517.

[4] N. Alon, G. Gutin and M. Krivelevich. Algorithms with large domination ratio, *J. Algorithms* 50: 118–131, 2004.

[5] N. Alon and J. Spencer, The Probabilistic Method, 2nd Edition, Wiley, 2000.

[6] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[7] J. Bang-Jensen and G. Gutin, Digraphs: Theory, Algorithms and Applications, Springer-Verlag, London, 2nd Ed., 2009.

[8] S. Böcker, F. Hüffner, A. Truss and M. Wahlström, A faster fixed-parameter approach to drawing binary tanglegrams. *IWPEC 2009*, Lect. Notes Comput. Sci. 5917: 38–49, 2009.

[9] H. L. Bodlaender, R.G. Downey, M.R. Fellows, and D. Hermelin, On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75(8): 423–434, 2009.

[10] H.L. Bodlaender, B.M.P. Jansen and S. Kratsch, Cross-Composition: A new technique for kernelization lower bounds. *STACS 2011*: 165–176.

[11] H.L. Bodlaender, S. Thomassé, and A. Yeo, Kernel bounds for disjoint cycles and disjoint paths, Theor. Comput. Sci. 412(35): 4570–4578, 2011.
A preliminary version was published in *ESA 2009*, Lect. Notes Comput. Sci. 5757: 635–646, 2009.

[12] B. Bollobás and A. Scott, Better bounds for max cut. In *Contemporary Combinatorics* (B. Bollobás, ed.), Springer, 2002.

[13] J.A. Bondy and U.S.R. Murty, *Graph Theory*. Springer, 2008.

[14] E. Boros and P.L. Hammer, Pseudo-Boolean optimization. Discrete Applied Math. 123(1-3): 155–225, 2002.

[15] P. Borwein. Computational excursions in analysis and number theory, Springer, New York, 2002.

[16] M. Charikar, V. Guruswami, and R. Manokaran, Every permutation CSP of arity 3 is approximation resistant. *Proc. Computational Complexity 2009*, 62–73.

[17] Y. Chen, J. Flum, and M. Müller, Lower bounds for kernelizations and other preprocessing procedures. *Proc. CiE 2009*, Lect. Notes Comput. Sci. 5635: 118–128, 2009.

[18] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM 55(5) (2008).

[19] C. Chiang, A. B. Kahng, S. Sinha, X. Xu, and A. Z. Zelikovsky. Fast and efficient bright-field AAPSM conflict detection and correction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(1):11–126, 2007.

[20] R. Crowston, G. Gutin and M. Jones, Directed Acyclic Subgraph Problem Parameterized above Poljak-Turzik Bound. *Proc. FSTTCS 2012*, LIPICS Vol. 18, 400–411.

[21] R. Crowston, G. Gutin, M. Jones, V. Raman and S. Saurabh, Parameterized Complexity of MaxSat Above Average. *Proc. LATIN 2012*, Lect. Notes Comput. Sci. 7256 (2012), 184–194.

[22] R. Crowston, G. Gutin, M. Jones, and A. Yeo, A new lower bound on the maximum number of satisfied clauses in Max-SAT and its algorithmic applications. *Algorithmica* 64 (2012), 56–68.

[23] R. Crowston, G. Gutin, and M. Jones, Note on Max Lin-2 above average. Inform. Proc. Lett. 110: 451–454, 2010.

[24] R. Crowston, G. Gutin, M. Jones, and G. Muciaccia. Maximum balanced subgraph problem parameterized above lower bound, *Proc. COCOON 2013*, to appear.

[25] R. Crowston, G. Gutin, M. Jones, E. J. Kim, and I. Ruzsa. Systems of linear equations over $\mathbb{F}_2$ and problems parameterized above average. *Proc. SWAT 2010*, Lect. Notes Comput. Sci. 6139 (2010), 164–175.

[26] R. Crowston, M. Fellows, G. Gutin, M. Jones, F. Rosamond, S. Thomassé and A. Yeo. Simultaneously Satisfying Linear Equations Over $\mathbb{F}_2$: MaxLin2 and Max-r-Lin2 Parameterized Above Average. *Proc. FSTTCS 2011*, LIPICS Vol. 13, 229–240.

[27] R. Crowston, G. Gutin, M. Jones, A. Yeo. Parameterized Complexity of Satisfying Almost All Linear Equations over $\mathbb{F}_2$, Theory of Computing Systems, June 2012

[28] R. Crowston, G. Gutin, M. Jones, V. Raman, S. Saurabh and A. Yeo. Fixed-Parameter Tractability of Satisfying Beyond the Number of Variables, *Algorithmica* October 2012.

[29] R. Crowston, M. Jones and M. Mnich, Max-Cut Parameterized above the Edwards-Erdős Bound, In *ICALP 2012*, Lect. Notes Comput. Sci. 7391 (2012) 242–253.

[30] M. Cygan, M. Pilipczuk, M. Pilipczuk and J. Wojtaszczyk. On Multiway Cut parameterized above lower bounds. *Proc. IPEC 2011*, Lect. Notes Comput. Sci. 7112 (2012), 1–12.

[31] B. DasGupta, G. A. Enciso, E. D. Sontag, and Y. Zhang. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. Biosystems 90(1):161–178, 2007.

[32] M. Dom, D. Lokshtanov and S. Saurabh, Incompressibility though Colors and IDs, *Proc. 36th ICALP, Part I*, Lect. Notes Comput. Sci. 5555: 378–389, 2009.

[33] R. G. Downey and M. R. Fellows. *Parameterized Complexity*, Springer, 1999.

[34] R. G. Downey, M. R. Fellows, A. Vardy and G. Whittle. The parameterized complexity of some fundamental problems in coding theory. *SIAM J. Comput.* 29(2): 545–570, 1999

[35] C.S. Edwards, An improved lower bound for the number of edges in a largest bipartite subgraph. *Recent Advances in Graph Theory, Proc. 2nd Czecholslovac Symp.*, Academia, Prague, 1995, 167–181.

[36] P. Erdős, On some extremal problems in graph theory. *Israel J. Math.* 3: 113–116, 1965.

[37] P. Erdős, A. Gyárfás and Y. Kohayakawa, The size of the largest bipartite subgraphs. *Discrete Math.* 117: 267–271, 1997.

[38] M. R. Fellows, Towards Fully Multivariate Algorithmics: Some New Results and Directions in Parameter Ecology. 20th International Workshop on Combinatorial Algorithms (IWOCA09), Lect. Notes Comput. Sci., 5874: 2–10, 2009.

[39] H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theor. Comput. Sci.*, 289(1):503–516, 2002.

[40] H. Fleischner and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(49), 2000.

[41] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, 2006.

[42] F. V. Fomin, D. Lokshtanov, V. Raman, S. Saurabh and B. V. R. Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, 2012.

[43] F.V. Fomin, D. Lokshtanov, N. Misra, G. Philip and S. Saurabh, Hitting forbidden minors: Approximation and kernelization. *Proc. STACS 2011*, LIPICS Vol. 9, 189–200.

[44] N. Gülpınar, G. Gutin, G. Mitra and A. Zverovitch, Extracting Pure Network Submatrices in Linear Programs Using Signed Graphs. Discrete Applied Mathematics 137 (2004) 359–372.

[45] G. Gutin, M. Jones, and A. Yeo. Kernels for below-upper-bound parameterizations of the hitting set and directed dominating set problems. *Theor. Comput. Sci.*, 412(41):5744–5751, 2011.

[46] G. Gutin, M. Jones and A. Yeo, A New Bound for 3-Satisfiable MaxSat and its Algorithmic Application. *Proc. FCT 2011*, Lect. Notes Comput. Sci. 6914 (2011), 138–147.

[47] G. Gutin, L. van Iersel, M. Mnich, and A. Yeo, Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables, J. Comput. System Sci. 78 (2012), 151–163.

[48] G. Gutin, E. J. Kim, M. Mnich, and A. Yeo. Betweenness parameterized above tight lower bound. J. Comput. Syst. Sci. 76 (2010), 872–878.

[49] G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. A probabilistic approach to problems parameterized above or below tight bounds. J. Comput. Sys. Sci. 77 (2011), 422–429.

[50] G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. A probabilistic approach to problems parameterized above tight lower bound. *Proc. IWPEC'09*, Lect. Notes Comput. Sci. 5917 (2009), 234–245.

[51] G. Gutin, A. Rafiey, S. Szeider, and A. Yeo. The linear arrangement problem parameterized above guaranteed value. Theory Comput. Syst., 41:521–538, 2007.

[52] G. Gutin, S. Szeider, and A. Yeo. Fixed-parameter complexity of minimum profile problems. Algorithmica, 52(2):133–152, 2008.

[53] G. Gutin and A. Yeo, Note on Maximal Bisection above Tight Lower Bound. *Inform. Proc. Lett.* 110: 966–969, 2010.

[54] G. Gutin and A. Yeo, Some Parameterized Problems on Digraphs. The Computer Journal 51 (2008) 363–371.

[55] G. Gutin and A. Zverovitch, Extracting pure network submatrices in linear programs using signed graphs, Part 2. Communications of DQM 6 (2003) 58–65.

[56] V. Guruswami, J. Håstad, R. Manokaran, P. Raghavendra, and M. Charikar, Beating the random ordering is hard: Every ordering CSP is approximation resistant. SIAM J. Comput. 40(3) (2011), 878–914.

[57] V. Guruswami, R. Manokaran, and P. Raghavendra, Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. *Proc. FOCS 2008*, 573–582.

[58] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression based fixed-parameter algorithms for feedback vertex set and edge bipartization. J. Comput. Syst. Sci., 72(8):1386–1396, 2006.

[59] F. Harary. On the notion of balance of a signed graph. Michigan Math. J., 2(2):143–146, 1953.

[60] G. H. Hardy and S. Ramanujan. Asymptotic formulae in combinatory analysis. *Proc. London Math. Soc.*, 17:75–115, 1918.

[61] J. Håstad, Some optimal inapproximability results. *J. ACM* 48: 798–859, 2001.

[62] J. Håstad and S. Venkatesh, On the advantage over a random assignment. *Random Structures & Algorithms* 25(2):117–149, 2004.

[63] M.A. Huang and K.J. Lieberherr, Implications of forbidden structures for extremal algorithmic problems. *Theoret. Comput. Sci.* 40: 195–210, 1985.

[64] F. Hüffner, N. Betzler, and R. Niedermeier. Optimal edge deletions for signed graph balancing. In 6th international Conf. on Experimental Algorithms (WEA'07), 297–310, 2007.

[65] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. J. Comput. Sys. Sci. 62 (2001), 367–375.

[66] R. Impagliazzo, R. Paturi and F. Zane. Which problems have strongly exponential complexity? J. Comput. Sys. Sci. 63 (2001), 512–530.

[67] S. Jukna, Extremal combinatorics: with applications in computer science, Springer, 2001.

[68] C. Käppeli and D. Scheder, Partial satisfaction of k-satisfiable formulas. *Electronic Notes in Discrete Math.* 29:497–501, (2007).

[69] S. Khot, On the power of unique 2-prover 1-round games. *Proc. STOC 2002*, 767–775.

[70] E.J. Kim and R. Williams, Improved Parameterized Algorithms for Above Average Constraint Satisfaction. *Proc. IPEC 2011*, Lect. Notes Comput. Sci. 7112 (2011), 118-131.

[71] H. Kleine Büning. On subclasses of minimal unsatisfiable formulas. *Discr. Appl. Math.*, 107(1-3):83–98, 2000.

[72] H. Kleine Büning and O. Kullmann, Minimal Unsatisfiability and Autarkies, Handbook of Satisfiability, chapter 11, 339–401.

[73] D.J. Kleitman, J.B. Shearer and D. Sturtevant. Intersection of k-element sets, Combinatorica, 1:381–384, 1981.

[74] B. Korte and J. Vygen, *Combinatorial Optimization: theory and algorithms*, 3rd Edition, Springer, 2006.

[75] S. Kratsch and M. Wahlström, Compression via Matroids: A Randomized Polynomial Kernel for Odd Cycle Transversal. *Proc. SODA 2012*: 94-103.

[76] D. Král, Locally satisfiable formulas. *Proc. SODA 2004*, 330–339, 2004.

[77] O. Kullmann. An application of matroid theory to the sat problem. In *IEEE Conference on Computational Complexity*, pages 116–124, 2000.

[78] O. Kullmann, Lean clause-sets: Generalizations of minimally unsatisfiable clause-sets, *Discr. Appl. Math.*,130:209-249, 2003.

[79] K.J. Lieberherr and E. Specker, Complexity of partial satisfaction. *J. ACM* 28(2):411-421, 1981.

[80] K.J. Lieberherr and E. Specker, Complexity of partial satisfaction, II. Tech. Report 293 of Dept. of EECS, Princeton Univ., 1982.

[81] D. Lokshtanov, *New Methods in Parameterized Algorithms and Complexity*, PhD thesis, Bergen, April 2009.

[82] D. Lokshtanov, D. Marx and S. Saurabh, Slightly superexponential parameterized problems, In *SODA 2011*, 760–776, 2011.

[83] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, S. Saurabh, Faster Parameterized Algorithms using Linear Programming, Arxiv preprint: `http://arxiv.org/abs/1203.0833v2`

[84] L. Lovász and M. D. Plummer. *Matching theory.* AMS Chelsea Publ., 2009.

[85] M. Mahajan and V. Raman. *Parameterizing above guaranteed values: MaxSat and MaxCut. J. Algorithms* 31(2):335–354, 1999.

[86] M. Mahajan, V. Raman, and S. Sikdar. Parameterizing MAX SNP problems above guaranteed values. *Proc. IWPEC'06*, Lect. Notes Comput. Sci. 4169 (2006), 38–49.

[87] M. Mahajan, V. Raman, and S. Sikdar. Parameterizing above or below guaranteed values. J. Computer System Sciences, 75(2):137–153, 2009. A preliminary version appeared in the 2nd IWPEC, *Lect. Notes Comput. Sci.* 4169:38–49, 2006.

[88] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discr. Appl. Math.* 10:287–295, 1985.

[89] M. Mnich, G. Philip, S. Saurabh, and O. Suchý, Beyond Max-Cut: λ-Extendible Properties Parameterized Above the Poljak-Turzík Bound. In *FSTTCS 2012*, LIPICS 18, 412–423, 2012.

[90] N. S. Narayanaswamy, V. Raman, M. S. Ramanujan and S. Saurabh. LP can be a cure for parameterized problems. *Proc. 29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, Leibniz International Proceedings in Informatics (LIPIcs), Volume 14, 2012, 338-349.

[91] R. Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press, 2006.

[92] R. Niedermeier, Reflections on Multivariate Algorithmics and Problem Parameterization. STACS 2010: 17–32.

[93] A. Nijenhuis and H. S. Wilf. Combinatorial Algorithms. *Academic Press, Inc.* 1978.

[94] R. O'Donnell, Some topics in analysis of Boolean functions. Technical report, ECCC Report TR08-055, 2008. Paper for an invited talk at STOC'08, `www.eccc.uni-trier.de/eccc-reports/2008/TR08-055/` .

[95] C. H. Papadimitriou and D. Wolfe. The complexity of facets resolved. *J. Comput. Syst. Sci.*, 37(1):2–13, 1988.

[96] S. Poljak and D. Turzík, A polynomial time heuristic for certain subgraph optimization problems with guaranteed worst case bound. Discrete Mathematics, 58 (1) (1986) 99–104.

[97] Arash Rafiey, Private Communication, Sept. 2011.

[98] V. Raman, M.S. Ramanujan and S. Saurabh, Paths, Flowers and Vertex Cover. *Proc. ESA 2011*, Lect. Notes Comput. Sci. 6942: 382–393, 2011.

[99] V. Raman and S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments. Theor. Comput. Sci., 351 (3) (2006) 446–458.

[100] I. Razgon and B. O'Sullivan. Almost 2-SAT is fixed-parameter tractable. J. Comput. Syst. Sci. 75(8):435–450, 2009.

[101] J. Spencer, Optimal ranking of tournaments. Networks 1 (1971) 135–138.

[102] A. Srinivasan. Improved approximations of packing and covering problems. In *STOC'95*, pages 268–276, 1995.

[103] S. Szeider, Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *J. Comput. Syst. Sci.*, 69(4):656–674, 2004.

[104] C.A. Tovey, A simplified satisfiability problem, Discr. Appl. Math. 8 (1984), 85–89.

[105] L. Trevisan. On local versus global satisfiability. *SIAM J. Discret. Math.* 17(4):541–547, 2004.

[106] Y. Villanger, P. Heggernes, C. Paul and J. A. Telle. Interval Completion Is Fixed Parameter Tractable. SIAM J. Comput., 38(5):2007–2020, 2009.

[107] D.B. West, *Introduction to Graph Theory*, 2nd Ed., Prentice Hall, 2001.

[108] R. de Wolf, A Brief Introduction to fourier analysis on the boolean cube, Theory Of Computing Library Graduate Surveys 1 (2008), 1–20, `http://theoryofcomputing.org` .

[109] M. Yannakakis. On the approximation of maximum satisfiability. *J. Algorithms*, 17:475–502, 1994.

[110] T. Zaslavsky. Bibliography of signed and gain graphs. Electronic Journal of Combinatorics, DS8, 1998.