MaxSAT with Absolute Value Functions: A Parameterized Perspective

Max Bannach
Institute for Theoretical Computer Science, Universität zu Lübeck, Germany

Pamela Fleischmann
Department of Computer Science, Christian-Albrechts-Universität zu Kiel, Germany

Malte Skambath
Department of Computer Science, Christian-Albrechts-Universität zu Kiel, Germany

Abstract

The natural generalization of the Boolean satisfiability problem to optimization problems is the task of determining the maximum number of clauses that can simultaneously be satisfied in a propositional formula in conjunctive normal form. In the *weighted maximum satisfiability problem* each clause has a positive weight and one seeks an assignment of maximum weight. The literature almost solely considers the case of *positive* weights. While the general case of the problem is only restricted slightly by this constraint, many special cases become trivial in the absence of *negative* weights. In this work we study the problem *with* negative weights and observe that the problem becomes computationally harder – which we formalize from a parameterized perspective in the sense that various variations of the problem become W[1]-hard if negative weights are present.

Allowing negative weights also introduces new variants of the problem: Instead of maximizing the sum of weights of satisfied clauses, we can maximize the *absolute value* of that sum. This turns out to be surprisingly expressive even restricted to *monotone* formulas in *disjunctive normal form* with at most *two literals* per clause. In contrast to the versions without the absolute value, however, we prove that these variants are *fixed-parameter tractable*. As technical contribution we present a kernelization for an auxiliary problem on hypergraphs in which we seek, given an edge-weighted hypergraph, an induced subgraph that maximizes the absolute value of the sum of edge-weights.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases parameterized complexity, kernelization, weighted maximum satisfiability, absolute value maximization

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.12

Related Version *Technical Report*: `https://arxiv.org/abs/2204.12614`

1 Introduction

Maximum satisfiability is the natural generalization of the satisfiability problem of propositional logic to optimization problems. In its decision version MAX-SAT, we seek a truth assignment for a given propositional formula in conjunctive normal form that satisfies at least $\alpha \in \mathbb{N}$ clauses. This problem is known to be NP-complete even in the restricted case that every clause contains at most two literals [16]. On the positive side, it is known that the problem can be decided in time $f(\alpha) \cdot n^c$ for some function $f\colon \mathbb{N} \to \mathbb{N}$ and constant $c \in \mathbb{N}$, which means that the problem is *fixed-parameter tractable* parameterized by α [26]. This also holds for the weighted version in which clauses have positive integer weights. In this paper we extend the previous scenario by allowing also *negative weights.* We denote the corresponding problem with MAX-CNF: decide for a given propositional formula with *arbitrary integer* weights whether there is an assignment of weight at least α. In the presence of negative weights, many non-trivial variants of the problem arise, for instance MAX-DNF (the propositional formula is in disjunctive normal form), MAX-MONOTONE-CNF

© Max Bannach, Pamela Fleischmann, and Malte Skambath;
licensed under Creative Commons License CC-BY 4.0

and MAX-MONOTONE-DNF (all literals in the formulas are positive), and ABS-CNF as well as ABS-DNF (maximize the *absolute value* of the sum of the satisfied clauses). Even an obscure combination such as ABS-MONOTONE-DNF becomes surprisingly expressive if negative weights are allowed (we show that this version is NP-hard restricted to formulas in which every clause contains at most two literals). This complexity jump (many of these versions are trivially in P if negative weights are *not* allowed) motivates a systematic study of the parameterized complexity of ABS-DNF and its variants. The main problem we investigate is defined as:

▶ **Problem 1** (p_α-d-ABS-DNF).

Instance: *A propositional formula ϕ in disjunctive normal form with clauses of size at most d, a target value $\alpha \in \mathbb{N}$, and a weight function $w\colon \mathrm{clauses}(\phi) \to \mathbb{Z}$.*
Parameter: α
Question: *Is there an assignment β such that $|\sum_{c \in \mathrm{clauses}(\phi), \beta \models c} w(c)| \geq \alpha$ (the absolute value of the sum of the weights of satisfied clauses is at least α)?*

We investigate the complexity of this problem by varying the status of α and d as parameter. In the notation above (α as index of p and d in the problem description) we consider α as a parameter and d as a fixed constant. But we also consider α *and* d to be parameters at the same time, or both to be constant, or to be neither a parameter nor a constant. An overview of the complexity of ABS-DNF can be found in Table 1.

🟨 **Table 1** Summary of our results for the maximum satisfiability problem over formulas in disjunctive normal form with negative weights and an absolute value target function (ABS-DNF). We distinguish how α (the weight of the sought solution) and d (maximum size of a clause) are interpreted (as *constant*, as *parameter*, or as *unbounded*). The columns correspond to d, while the rows represent α. Upper bounds are highlighted in green and lower bounds in red.

$\alpha \diagdown d$	*constant*	*parameter*	*unbounded*
constant	P (Corollary 25)	FPT (Corollary 24)	unknown
parameter	FPT (Theorem 3)	FPT (Corollary 24)	W[1]-hard (Corollary 12)
unbounded	NP-hard (Corollary 9)	para-NP-hard (Corollary 10)	NP-hard (Corollary 9)

As mentioned before, we will also study other variations of the problem, which we obtain by replacing the DNF by a CNF, by forcing the formula to be monotone, or by replacing the absolute value function by a normal sum. It turns out that without the absolute value function, many natural versions of the problem become W[1]-hard in the presence of negative weights. However, the problems become tractable if the aim is to maximize the absolute value of the target sum as they than can all be fpt-reduced to p_α-d-ABS-MONOTONE-DNF, which in turn is equivalent to the following hypergraph problem:

▶ **Problem 2** (p_α-d-UNBALANCED-SUBGRAPH).

Instance: *A d-hypergraph $H = (V, E)$, a weight function $w\colon E \to \mathbb{Z}$, and a number $\alpha \in \mathbb{N}$.*
Parameter: α
Question: *Is there a set $X \subseteq V$ with $|w[X]| \geq \alpha$.*

In words, we seek a subset X of the vertices of an edge-weighted hypergraph such that the absolute value of $w[X] = \sum_{e \in E[X]} w(e)$ is maximized. Here $E[X]$ are the hyperedges induced by the set X. Intuitively, that means we have to find a subgraph that contains either more positive than negative edges or more negative edges than positive ones. Thus, we seek an *unbalanced* subgraph.

Our Contribution I: Hardness Results. We present hardness results for the parameterized and non-parameterized variants of MAX-DNF and ABS-DNF (both with negative weights). In particular, we prove that MAX-DNF and ABS-DNF are NP-hard when restricted to monotone formulas with at most two literals per clause. However, the problems differ from a parameterized perspective: p_α-d-MAX-MONOTONE-DNF is W[1]-hard for all $d \geq 2$, while p_α-d-ABS-DNF $\in$ FPT. We also prove a similar result for the CNF-versions: p_α-d-MAX-MONOTONE-CNF is W[1]-hard for all $d \geq 2$, but p_α-d-ABS-CNF $\in$ FPT.

Our Contribution II: Satisfiability Based Optimization with Absolute Value Function. We derive a kernelization algorithm for p_α-d-UNBALANCED-SUBGRAPH, which implies that the problem is in FPT parameterized by α. Using an fpt-reduction we show:

▶ **Theorem 3.** p_α-d-ABS-DNF $\in$ FPT

Our Contribution III: Absolute Integer Optimization. Theorem 3 can be generalized to an *absolute integer optimization problem.* Given a multivariate polynomial $p\colon \mathbb{Z}^n \to \mathbb{Z}$ with integer coefficients and intervals $[(b_{min})_i; (b_{max})_i]$ for every $i \in \{1, \ldots, n\}$, the task is to find a solution $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ with $(b_{min})_i \leq x_i \leq (b_{max})_i$ such that the non-linear objective function $|p(x_1, \ldots, x_n)|$ is maximized. With the value of the sought solution α and the maximal degree d of the polynomial, we prove:

▶ **Theorem 4.** $\mathrm{p}_{\alpha,d}$-ABS-IO $\in$ FPT

We emphasize that there are no restrictions on the number of variables or on the size of an interval $[(b_{min})_i; (b_{max})_i]$. There may be even infinitely large intervals like $[-\infty; \infty]$.

Related Work. Optimization problems based on satisfiability or constraint-satisfaction problems are usually NP-complete [23, 31]. The parameterized complexity of these problems is, thus, an active field of research [17, 18, 21, 32]. An overview over the current trends can be found in [8, 30]. In a propositional formula in conjunctive normal form, we can always satisfy at least $(1 - 2^{-d})m$ clauses if m is the number of clauses and d is the size of these clauses [7, Chapter 9.2]. Therefore, the problem is fixed-parameter tractable when parameterized by the solution size α. It is known that the problem remains in FPT parameterized above this lower bound, i. e., the parameter is the distance between $(1-2^{-d})m$ and the number of clauses that can actually be satisfied simultaneously [1]. The currently fastest algorithm for parameter α runs in time $O^*(1.3248^\alpha)$ and improving the base is an active field of research [3].

Another parameterized approach to (non-optimizing) satisfiability problems are *backdoors*, which are small sets of variables such that assigning values to these variables yields an easy formula, e. g., a KROM- or HORN-formula [5, 28]. Unfortunately, MAX-CNF remains NP-hard on such formulas, making the approach less appealing for optimization problems.

A third line of research on parameterized algorithms for satisfiability problems focuses on *structural parameters*, i. e., graph theoretic properties of a graph representation of the formula. Depending on the details of that representation, various tractability and intractability results can be derived for parameters such as the *treewidth* of the corresponding graph [29, 30]. Such algorithms usually can be obtained from general frameworks such as Courcelle's Theorem and carry over to the optimization versions [4]. Generalizing this approach to a broader range of graphs is an active field of research, see for instance [15] or [2, Chapter 17].

A related problem is MAX-LIN2 [1, 6, 27]: given variables $x_1, \ldots, x_n$ and a set of equations $\Pi_{i \in I} x_i = b$ for $b \in \{-1, 1\}$ and $I \subseteq \{1, \ldots, n\}$, find an assignment to $\{-1, 1\}$ that satisfies as many equations as possible. This can be seen as maximizing *parity constraints*, in contrast to *or constraints* (MAX-CNF) or *and constraints* (MAX-DNF). The parameterized complexity of *linear* integer programming is well understood, see for instance the famous result of Lenstra [25] or recent developments in algorithms based on structural parameters [9, 14, 22, 24]. There are also recent results on *non-linear* programs, see for instance [10, 19, 20]. For separable convex functions, fixed-parameter tractability results are known for structural parameters depending on the constraint matrix [11]. There are also results that directly cover absolute values of polynomials – for instance, for maximizing convex functions, which include absolute values of convex polynomials [20].

Structure of this Work. After some brief preliminaries in Section 2, we present lower bounds for various versions of ABS-DNF and related problems in Section 3. Afterwards, we formulate our main result – an FPT-algorithm for p_α-d-ABS-DNF – in Section 4. We also present the underlying machinery, i. e., a kernelization-algorithm for p_α-d-UNBALANCED-SUBGRAPH. Section 5 concludes with further applications and an outlook is presented in Section 6. Due to space constraints, some proofs are only available in the technical report.

2 Preliminaries

A *parameterized problem* is a set $Q \subseteq \Sigma^* \times \mathbb{N}$, where an instance (w, k) consists of a word w and a *parameter* k. We denote parameterized problems with a preceding "p-" with an index that indicates what the parameter is – for instance, p_k-VERTEX-COVER denotes the well-known vertex cover problem parameterized by the solution size k. A parameterized problem is *fixed-parameter tractable* (*fpt* or it is in FPT) if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$ such that we can decide $(w, k) \overset{?}{\in} Q$ in time $f(k) \cdot |w|^c$. A *size-g kernelization* for a parameterized problem Q and a computable function $g\colon \mathbb{N} \to \mathbb{N}$ is a polynomial-time algorithm that, on input of an instance (w, k), outputs an instance (w', k') such that $(w, k) \in Q \Leftrightarrow (w', k') \in Q$ and $|w'| + k' \leq g(k)$. The output of the algorithm is called a *kernel.* If g is a polynomial then the kernel is called a *polynomial kernel.* It is well-known that a decidable problem is in FPT iff it admits a kernelization [13, Theorem 1.39].

A *d-hypergraph* $H = (V, E)$ consists of a set V of vertices and a set E of edges with $e \subseteq V$ and $|e| \leq d$ for all $e \in E$. A hypergraph is *d-uniform* if all edges have size *exactly* d. For $d = 2$, they are just *graphs.* The *neighborhood* of a vertex v is the set $N(v) = \{\, w \mid (\exists e \in E)(v, w \in e) \,\}$ and its degree $\deg(v)$ is the number of edges containing it. The *link* of a set $c \subseteq V$ is defined as $\mathrm{link}(c) = \{\, e \mid e \in E \text{ and } c \subsetneq e \,\}$. A *subedge* of some hyperedge $e \in E$ is any subset $s \subseteq e$. We use $\Delta(H) = \max_{v \in V} \deg(v)$ (or just Δ, if H is clear from the context) to denote the *maximum degree of* H. For $X \subseteq V$ and $w\colon E \to \mathbb{Z}$, let $E[X] = \{\, e \in E \mid e \subseteq X \,\}$ denote the edges of the *induced hypergraph* $H[X] = (X, E[X])$ over X, and define its *weight* as $w[X] = \sum_{e \in E[X]} w(e)$.

We need only little terminology from propositional logic. A formula ϕ is in *disjunctive normal form* if it is a disjunction of conjunctions, e. g., $\psi \equiv (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$. It is in *conjunctive normal form* if it is a conjunction of disjunctions, e. g., $\chi \equiv (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3)$. We refer to the variables of ϕ with $\mathrm{vars}(\phi)$, and we call the conjunctions in a formula in disjunctive normal form and the disjunctions in a formula in conjunctive normal form *clauses.* These objects are addressed with $\mathrm{clauses}(\phi)$. For instance, $\mathrm{vars}(\psi) = \mathrm{vars}(\chi) = \{x_1, x_2, x_3\}$, $\mathrm{clauses}(\psi) = \{(x_1, x_2), (x_2, x_3)\}$, and $\mathrm{clauses}(\chi) = \{(x_1, x_2), (x_2, \neg x_3)\}$. A *literal* is a variable (then it is *positive*) or its negation (then it is *negative*). Formulas that contain only positive literals are *monotone*, e. g., ψ is monotone but χ is not.

3 Lower Bounds for MaxSAT and AbsSAT Based Optimization

The problems MAX-DNF and ABS-DNF may seem similar on first sight. After all, if there are no negative weights they are clearly equivalent. However, if negative weights are present, a solution of ABS-DNF may either construct a sum of at least α or of at most $-\alpha$; in contrast, a solution for MAX-DNF does not have such freedom and has to find a weight-α solution. This results in an interesting complexity gap: p_α-d-MAX-MONOTONE-DNF is W[1]-hard for $d \geq 2$ while p_α-ABS-DNF is only W[1]-hard if d is unbounded (recall that d is the size of the clauses and that in p_α-d-MAX-MONOTONE-DNF the size of every clause is bounded by a constant d). We prove the hardness results in this section and complement them with a proof of p_α-d-ABS-DNF $\in$ FPT in the next section.

We first consider p_α-d-MAX-MONOTONE-DNF. This problem is W[1]-hard by a reduction from the independent set problem (given a graph and an integer k, decide whether there is a set of at least k pairwise non-adjacent vertices):

▶ **Lemma 5.** *p_α-d-MAX-MONOTONE-DNF is W[1]-hard for $d \geq 2$.*

Proof. Let $G = (V, E)$ and $k \in \mathbb{N}$ be an instance of p_k-INDEPENDENT-SET. We build a formula ϕ that contains a variable x_v for every $v \in V$:

$$\phi \equiv \bigvee_{v\in V} \underset{\substack{\downarrow \\ 1}}{(x_v)} \vee \bigvee_{\{v,w\}\in E} \underset{\substack{\downarrow \\ -1}}{(x_v \wedge x_w)}.$$

The red labeling shows the corresponding weight function w. Clearly, a size-k independent set translates into an assignment of weight k. For the other direction consider an assignment of weight k. We can assume that the assignment satisfies only positively weighted clauses. Otherwise we can flip one variable of a satisfied and negatively weighted clause. This does not decrease the weight because there is at most one positively weighted clause containing this variable. Since k variables are set to *true* and no negatively weighted clauses are satisfied, the assignment translates back to an independent set. ◀

▶ **Corollary 6.** *p_α-d-MAX-DNF is W[1]-hard for $d \geq 2$.*

On the other hand, the absolute value version of the problem remains intractable in the classical sense, which motivates our parameterized perspective. In detail, the (unparameterized) problem d-ABS-DNF is NP-hard by a reduction from INDEPENDENT-SET. Note that in contrast to the previous lemma, the reduction in the following is not parameter-preserving. Thus, the theorem does *not* imply that p_α-d-ABS-MONOTONE-DNF is W[1]-hard.

▶ **Theorem 7.** *d-ABS-MONOTONE-DNF is NP-hard for $d \geq 2$.*

Proof. We reduce from INDEPENDENT-SET and let $G = (V, E)$ with $k \in \mathbb{N}$ be a corresponding instance. Construct the following formula ϕ that contains four variables $v_1^+, v_2^+, v_1^-, v_2^-$ for every vertex $v \in V$ (the red arrows illustrate the weight function):

$$\phi \equiv \bigvee_{v\in V} \underset{\substack{\downarrow \\ -1}}{(v_1^+ \wedge v_2^+)} \vee \underset{\substack{\downarrow \\ 1}}{(v_1^- \wedge v_2^-)} \vee \bigvee_{\{v,w\}\in E} \underset{\substack{\downarrow \\ 1}}{(v_1^+ \wedge w_1^+)} \vee \underset{\substack{\downarrow \\ -1}}{(v_1^- \wedge w_1^-)}.$$

The intuition (also illustrated in Example 8) behind the formula is as follows: Think of two weighted copies of G called G^+ and G^- such that all edges in G^+ have weight $+1$ and all edges in G^- have weight -1. This is the second part of the formula. Observe that by taking either all edges of G^+ or all edges of G^-, a solution would always have absolute value $|E|$.

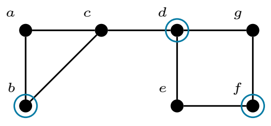
To encode the independent set problem, we introduce an extra gain for every vertex, these are the variables v_2^+ and v_2^-. Observe that in the first part of ϕ, we can add -1 or $+1$ to the solution by setting $v_1^{+/-}$ and $v_2^{+/-}$ to *true* at the same time. Finally, note that the signs in the first part of the formula are exactly the opposite of the signs in the second part. We claim that G has a size-k independent set iff ϕ has a solution of weight $k+|E|$. Let S be a size-k independent set in G, then the following assignment has an absolute value of $k+|E|$:

$$\beta(v_i^\sigma) = \begin{cases} \mathit{true} & \text{if } (v \in S \text{ and } \sigma = -) \text{ or } (\sigma = + \text{ and } i = 1); \\ \mathit{false} & \text{else.} \end{cases}$$

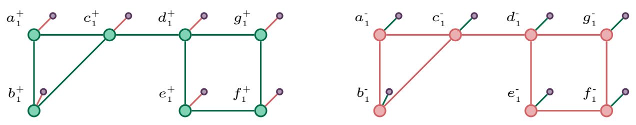
For the other direction let β be an assignment that achieves an absolute value $|\nu|$ of at least $k+|E|$. Assume for simplicity that $\nu > 0$ (the case $\nu < 0$ is symmetric). We may assume $\beta(v_1^+) = \mathit{true}$ and $\beta(v_2^+) = \mathit{false}$ – otherwise we can increase the solution by modifying the assignment locally. Observe that by setting only variables v_i^+ to *true*, we have $\nu \leq |E|$ and, since we assumed $|\nu| \geq k+|E|$, β sets further variables v_j^- to *true*. We may assume $\beta(v_1^-) = \beta(v_2^-)$ for all $v \in V$, as setting both variables to *true* is the only way to increase ν (the clauses of the form $(v_1^- \wedge w_1^-)$ all have a negative weight and we assumed $\nu > 0$).

Let $\{v, w\} \in E$ be an edge such that $\beta(v_1^-) = \beta(w_1^-) = \mathit{true}$. Observe that changing $\beta(w_1^-)$ to $\beta(w_1^-) = \mathit{false}$ does *not* decrease the value of the solution (it decreases the number of satisfied clauses with negative weight by exactly one, as the positively weighted clause $(w_1^- \wedge w_2^-)$ is not satisfied anymore; but it increases the number of satisfied clauses with positive weight by at least one as the negatively weighted clause $(v_1^- \wedge w_1^-)$ is also no longer satisfied). Hence, we may adapt β such that $S = \{\, v \mid \beta(v_1^-) = \mathit{true} \,\}$ is an independent set. Since the absolute value of β is at least $|E| + k$ and since all clauses containing variables of the form v_i^+ can contribute at most $|E|$ to this sum, we have $|S| \geq k$. ◀

▶ **Example 8.** Let us illustrate the reduction used within the theorem with an example. Consider the following graph G with an independent set of size three:



Within the reduction, we virtually generate the following edge-weighted graph that contains two copies of G, namely G^+ and G^-. In both copies, every vertex has a mirror vertex attached to it. Edges have either a positive weight of $+1$ or a negative weight of -1:



The reduction continues by defining a weighted formula that contains a clause for every edge with the same weight as that edge, e. g., we would have $(a_1^+ \wedge b_1^+)$ of weight $+1$ and $(a_1^- \wedge b_1^-)$ of weight -1. By selecting from one copy (i. e., setting the corresponding variables to *true*) all vertices except the mirrors, we obtain an assignment of weight $\pm|E|$; selecting the independent set together with its mirrors in the other copy provides another $\pm k$, yielding a solution of absolute value $|E|+k$. In this example we could take all green vertices (providing a weight of $+8$) and $\{b_1^-, d_1^-, f_1^-\}$ together with their mirrors (for another $+3$); yielding a solution with absolute value 11. ⌟

▶ **Corollary 9.** *d-ABS-DNF is NP-hard for $d \geq 2$.*

▶ **Corollary 10.** p_d-ABS-DNF *is* para-NP*-hard.*

Due to the previous theorem, we cannot hope to achieve parameterized tractability with respect to the sole parameter d. The following lemma shows that this is also not possible for the parameter α. In the upcoming sections, we thus rely on the combined parameter $\alpha + d$.

▶ **Lemma 11.** p_α-ABS-MONOTONE-DNF *is* W[1]*-hard.*

Proof. Let $G = (V, E)$ and $k \in \mathbb{N}$ be an instance of p_k-INDEPENDENT-SET. Consider the following formula ϕ that contains a variable x_v for every vertex $v \in V$:

$$\phi \equiv \bigvee_{v \in V} \left(\neg x_v \wedge \bigwedge_{\{v,w\} \in E} x_w \right).$$

Let $\alpha = k$ and w be a weight function that maps all clauses of ϕ to 1. We show that (G, k) is a yes-instance of INDEPENDENT-SET iff (ϕ, w, α) is a yes-instance of ABS-DNF. For the first direction let $I \subseteq V$ be a size-k independent set and consider the assignment:

$$\beta(x_v) = \begin{cases} \mathit{false} & \text{if } v \in I; \\ \mathit{true} & \text{otherwise.} \end{cases}$$

Since I is an independent set we have $\beta(x_w) = \mathit{true}$ for all vertices w that have a neighbor $v \in I$. Therefore, β satisfies every clause that corresponds to a vertex of I and, hence, β is an assignment of weight at least k.

For the other direction let β be an assignment of weight at least k, i. e., one that satisfies at least k clauses. For each of these clauses there is a unique vertex v such that x_v appears negatively only in that clause and, thus, at least k variables must be assigned to *false* by β. Furthermore, for all neighbors w of v we have $\beta(x_w) = \mathit{true}$ by the second part of the clause. We conclude that $\{\, v \mid \beta(x_v) = \mathit{false} \,\}$ is an independent set of size at least k.

The formula ϕ is not monotone, however, in the proof of Lemma 17 we describe how an arbitrary instance can be turned into an equivalent one with a monotone formula. In this case we obtain the following monotone formula ϕ', which (instead of ϕ) can be constructed directly for this reduction.

$$\phi' \equiv \bigvee_{v \in V} \left(\underset{\overset{\downarrow}{1}}{\bigwedge_{\{v,w\} \in E}} x_w \right) \vee \bigvee_{v \in V} \left(x_v \wedge \underset{\overset{\downarrow}{-1}}{\bigwedge_{\{v,w\} \in E}} x_w \right).$$

The red labeling shows the corresponding weight function w'. As we describe in the proof of Lemma 17 the instances (ϕ, w, k) and (ϕ', w', k) are equivalent. This completes the proof. ◀

▶ **Corollary 12.** p_α-ABS-DNF *is* W[1]*-hard.*

3.1 From Disjunctive Normal Form to Conjunctive Normal Form

So far, we have seen that the parameterized DNF maximization variants are W[1]-hard. We can use this to show that the same holds also for the conjunctive normal forms:

▶ **Lemma 13.** p_α-d-MAX-MONOTONE-CNF *is* W[1]*-hard for* $d \geq 2$*.*

▶ **Corollary 14.** p_α-d-MAX-CNF *is* W[1]*-hard for* $d \geq 2$*.*

3.2 Finding an Assignment with a Certain Weight

We finish this section with a variant of the problem in which we seek an assignment of a certain absolute value. In detail, $\mathrm{p}_{\alpha,d}$-EXACT-ABS-MONOTONE-DNF asks, given a weighted propositional formula in disjunctive normal form, whether there is an assignment such that the absolute value of the weighted sum is *exactly* α (and not *at least* α).

▶ **Lemma 15.** *d-EXACT-ABS-MONOTONE-DNF is NP-hard for any constant $d \geq 2$ and $\alpha = 0$.*

Proof. Since p_α-d-MAX-MONOTONE-DNF is W[1]-hard by Lemma 5, it is not surprising that d-EXACT-MAX-MONOTONE-DNF is NP-hard. We just use the same polynomial-time-computable reduction, which is possible since the independent set problem is monotone in the sense that the existence of a solution of size $\geq k$ implies the existence of an independent set of size $= k$.

We can turn a d-EXACT-MAX-MONOTONE-DNF instance into a d-EXACT-ABS-MONOTONE-DNF instance: Just add the empty clause with weight $-\alpha$ and set the target value to 0. Every assignment of weight α in the original instance will have weight 0, as the empty clause is always satisfied. On the other hand, any weight-0 assignment in the new formula has to satisfy clauses of weight exactly α to compensate for the empty clause. ◀

▶ **Corollary 16.** *p_α-d-EXACT-ABS-MONOTONE-DNF is para-NP-hard.*

Proof. A parameterized problem is para-NP-hard if there are finitely many slices whose union is NP-hard [13, Theorem 2.14]. The claim follows as already the slice $\alpha = 0$ alone is NP-hard by Lemma 15. ◀

4 Satisfiability Based Optimization with Absolute Value Function

In the previous section we showed that various SAT-based optimization problems are presumably not in FPT with respect to the natural parameter α. There are various ways to deal with these negative results. For instance, we could restrict ourselves to structured instances – such as formulas of bounded incidence-treewidth [8]. For the absolute version of the problem, we fortunately do not need such a strong structural parameter. Instead, we provide a reduction to an auxiliary hypergraph problem for which we present a polynomial kernel. This results in the main theorem of this section:

▶ **Theorem 3.** p_α-d-ABS-DNF $\in$ FPT

It is well-known that various optimization problems over CNFs and DNFs are equal to problems on hypergraphs. For instance, it is easy to see that the question of whether a given *monotone* d-CNF can be satisfied by setting at most k variables to *true* equals the hitting set problem on d-hypergraphs. Notice that "monotone" here is a crucial key word, as this property allows us to encode clauses as edges. Such an approach does *not* work directly for p_α-d-MAX-DNF. Fortunately, p_α-d-ABS-DNF reduces to its monotone version:

▶ **Lemma 17.** p_α-d-ABS-DNF $\leq$ p_α-d-ABS-MONOTONE-DNF

Proof. Let ϕ be a d-DNF and $w\colon \mathrm{clauses}(\phi) \to \mathbb{N}$ be a weight function on its clauses. We argue how we can remove one occurrence of a negative literal – an iterated application of this idea then leads to the claim.

Let $c \in \text{clauses}(\phi)$ be a clause containing a negative literal $\overline{x}$. We remove c from ϕ and add two new clauses: c^+ and c^-. The clause c^+ equals c without $\overline{x}$, and we set $c^- = c^+ \wedge x$. Define $w(c^+) = w(c)$ and $w(c^-) = -w(c)$, and observe that an assignment that satisfies c does satisfy c^+ but not c^- and, thus, yields the same value. On the other hand, every assignment that satisfies c^- does also satisfy c^+ and, hence, the values cancel each other. ◀

Given a monotone DNF and a weight function on its clauses, we can encode the problem p$_\alpha$-d-ABS-MONOTONE-DNF as a hypergraph by introducing one vertex per variable and by representing clauses as edges. The task then is to identify a set of vertices such that the absolute value of the total weight of the edges appearing in the subgraph induced by the vertices is as large as possible. Formally this is Problem 2 from the introduction.

▶ **Corollary 18.** p$_\alpha$-d-ABS-DNF $\leq$ p$_\alpha$-d-UNBALANCED-SUBGRAPH

4.1 A Polynomial Kernel for Unbalanced Subgraph

In this section we develop a polynomial kernel for p$_\alpha$-d-UNBALANCED-SUBGRAPH. The kernelization consists of a collection of *reduction rules* which we just call *rules.* Every rule obtains as input an instance of p$_\alpha$-d-UNBALANCED-SUBGRAPH and either (i) is not applicable, or (ii) outputs an equivalent smaller instance. The rules are numbered and we assume that whenever we invoke a rule, all rules with smaller numbers have already been applied exhaustively. It will always be rather easy to show that the rules can be implemented in polynomial time, so we only prove that they are safe. To get started, we consider the following simple rules, where Δ is the maximum vertex degree in the hypergraph:

▶ **Rule 1.** *If H contains an isolated vertex v, delete v.*

▶ **Rule 2.** *If H contains an edge e with $w(e) = 0$, delete e.*

▶ **Rule 3.** *If H has at least $2\alpha \cdot d^3\Delta^2$ vertices, reduce to a trivial yes-instance.*

▶ **Lemma 19.** *Rule 1–3 are safe.*

Proof. For the first two rules observe that neither isolated vertices nor weight-0 edges can contribute to any solution and, hence, may be discarded.

For Rule 3 we argue that H contains a set $X \subseteq V$ with $|w[X]| \geq \alpha$. Consider a maximal set $M \subseteq E$ such that (i) all edges in M are pairwise disjoint and (ii) $H\bigl[\cup_{e\in M} e\bigr] = \bigl(\cup_{e\in M} e, M\bigr)$, that is, M is a set packing that induces itself. Such a set always exists and can also be computed as the following greedy algorithm shows: Start with $M = \emptyset$ and repeatedly pick a remaining nonempty edge e from H that is no superset of any edge in H (except $\emptyset$) and add it to M. Since Rule 1 has already removed isolated vertices, there must be such an edge as long as V is not empty. After picking an edge e, remove the set E_e of all edges that intersect e. Observe that $|E_e| \leq d\Delta$. Clearly, if we just remove E_e, the algorithm would already generate a maximal set packing – however, it would not necessarily be self-induced. Let therefore $V_e = \{\, v \in V \mid (\exists e' \in E_e)(v \in e') \,\}$ be the set of vertices covered by the to-be-removed edges. Since every edge has size at most d, we have $|V_e| \leq d^2\Delta$. To ensure that M will be self-induced, we remove all edges that contain a vertex of V_e – which are at most $d^2\Delta^2$ edges as H has maximum degree Δ.

For every edge that is picked to be in M, at most $d^2\Delta^2$ edges are removed from $|E|$. Therefore we have $|M| \geq |E|/\bigl(d^2\Delta^2\bigr)$. Furthermore, we have $|E| \geq |V|/d$ because Rule 1 has already removed isolated vertices. Given that $|V| \geq 2\alpha \cdot d^3\Delta^2$, we deduce $|M| \geq 2\alpha$. There are at least α edges in M that have the same sign (no edge has a zero-weight by Rule 2). Therefore, the set X of vertices induced by the larger set satisfies $|w[X]| \geq \alpha$. ◀

The rules yield a kernel for the combined parameter $\alpha+\Delta$, which means that an exhaustive application of these rules leaves an instance of size bounded in α and Δ.

▶ **Corollary 20.** $\mathrm{p}_{\alpha,\Delta}$-$d$-UNBALANCED-SUBGRAPH *has a kernel with $O(\alpha \cdot \Delta^2)$ vertices.*

If we only consider $\alpha + d$ as parameter, rules 1–3 are not applicable, and the graph is still large, we know that the input graph has no isolated vertices, all edges have a non-zero weight, and we know that there is at least one vertex of high degree. Our final reduction rule states that in this case we can as well reduce to a trivial yes-instance, since the high degree vertex induces a solution. Intuitively, this is similar to an application of the well-known Sunflower Lemma [12], which states that large enough hypergraphs contain a set of hyperedges that pairwise all intersect in the same set of vertices (i. e., the "hypergraph version" of a high-degree vertex). Unfortunately, a direct application of the lemma is tricky as other hyperedges may intersect arbitrarily with such sunflowers. To circumnavigate this difficulties, we need objects that are more like a *self-induced sunflower.* This leads to the following reduction rule; details of these concepts are postponed to the proof of Lemma 21.

▶ **Rule 4.** *Let $g(i) = (i^i 2\alpha \cdot 2^{2^d})^{2^i-1}$ for $i > 0$ and $g(0) = 1$. If H contains a subedge $c \subsetneq e \in E$ such that (i) $|\operatorname{link}(c)| \geq g(i)$ where $i = d - |c|$ and (ii) for every superset $f \supsetneq c$ it holds that $|\operatorname{link}(f)| < g(j)$ where $j = d - |f|$, then reduce to a trivial yes-instance.*

Before we prove the safeness of the rule in the general case, let us quickly sketch why the rule is correct for 2-uniform hypergraphs, i. e., normal graphs. A vertex v with $\deg(v) \geq 4\alpha$ is incident to at least 2α edges of the same sign. For the sake of an example let there be a set $P(v) \subseteq N(v)$ with $|P(v)| \geq 2\alpha$ and $w(\{v, u\}) > 0$ for all $u \in P(v)$. Either $P(v)$ is a solution (i. e., $|w[P(v)]| > \alpha$) or $P(v) \cup \{v\}$ is a solution (since $w[P(v) \cup \{v\}] \geq 2\alpha - |w[P(v)]| \geq \alpha$).

▶ **Lemma 21.** *Rule 4 is safe.*

Proof. Let $c \subseteq V$ be a set for which the rule is applicable. We argue that H contains a set $X \subseteq V$ with $|w[X]| \geq \alpha$. Let $i = d - |c|$, $E(c) = \operatorname{link}(c) = \{e \in E \mid c \subsetneq e\}$, $N(c) = \bigcup E(c) \setminus c$, and $\deg_{E(c)}(v) = |\{e \in E(c) \mid v \in e\}|$. We can assume that $|c| < d$ and $i > 0$ because otherwise $|\operatorname{link}(c)| = 0 < g(i)$. Since $|\operatorname{link}(f)| < g(j)$ holds for any superset $f \supsetneq c$ with $j = d - |f|$, it also holds that $\deg_{E(c)}(v) \leq g(i-1)$ for every vertex $v \in N(c)$. Otherwise we have a contradicion with $f = c \cup \{v\}$.

Consider a maximal set $M \subseteq E(c)$ such that (i) all edges in M contain c as a subedge, (ii) c is the intersection of any pair of different edges in M, and (iii) for every edge $e \in E(c)$ it holds that either $e \in M$, or $e \nsubseteq \left(\bigcup_{e' \in M} e'\right)$, that is, M is a set of edges that induces only itself in $H_c = (V, E(c))$. By (i) and (ii) M is also known as a *sunflower* with *core c*. Such a set always exists as the following greedy algorithm shows: Start with $M = \emptyset$ and observe that all three properties hold. Repeatedly pick an edge $e \in E(c)$ that has no smaller subedge in $E(c)$ and add it to M as long as the properties are preserved.

We will use M to identify the soughted set X. For this, we require a lower bound for the size of M. Therefore, let us carefully analyze which edges cannot be choosen after an edge has been added to M. After picking an edge $e \in E(c)$, we can remove the whole set $E_e \subseteq E(c)$ of all edges that intersect e not only in c (a proper superset of c) because picking one of those edges contradicts property (ii). There may be also an edge $e' \in E(c) \setminus E_e$ that has not been added yet, but for which there is another edge $e'' \in E(c) \setminus M$ with $e'' \in \bigcup M \cup \{e'\}$. This implies that e' may not be added to M to preserve property (iii). Furthermore, there must be an edge $e \in M$ such that e' intersects either e or some edge in E_e outside in the core. Let $E'_e \subseteq E(c)$ be the set of all edges that intersect E_e not only in c. It follows that any edge in $E(c)$ will either be inserted to M, or is in E_e or E'_e for an edge $e \in M$. Observe that $|E_e| \leq ig(i-1)$ because $e \setminus c$ contains at most i vertices that are in at most $g(i-1)-1$ edges from $E(c)$. For the same reason, we have $|E'_e| \leq |E_e| \cdot (i-1) \cdot g(i-1)$. Therefore, $|\{e\} \cup E_e \cup E'_e| \leq i^2(g(i-1))^2$ edges are removed for every edge in M. This implies $|M| \geq |E(c)|/\bigl(i^2 g(i-1)^2\bigr)$. Given that $E(c) \geq g(i)$, we deduce $|M| \geq 2\alpha \cdot 2^{2^d}$: For $i = 1$, we have $|M| \geq |E(c)|$ because of $g(0) = 1$. For $i > 1$ we use that $x^{2^i-1}/x^{2\cdot 2^{i-1}-2} = x$ and $g(i) = x^{2^i-1}$ for $x = 2\alpha \cdot 2^{2^d}$.

We show that we can find a set $X \subseteq \bigcup_{e\in M} e$ with $|w[X]| \geq \alpha$. Let M^+ denote all positively weighted edges in M and let M^- denote the other ones. Let us consider three cases based on the size of the core.

First Case ($|c| = 0$). We have $c = \emptyset$ and, without loss of generality, let $w[\emptyset] = |w[\emptyset]| \geq 0$. Otherwise we could flip the sign of all weights. Note that $E(\emptyset) = E \setminus \{\emptyset\}$. By Property (iii), any subset of $\bigcup M$ induces only edges in $M \cup \{\emptyset\}$. It follows that $w[\bigcup M^+] = |M^+| + w[\emptyset]$. We are done for $w[\bigcup M^+]| \geq \alpha$, otherwise we obtain

$$|w[\bigcup M^-]| = |M^-| - w[\emptyset] = |M| - (|M^+| + w[\emptyset]) > |M| - \alpha.$$

The claim follows with $|M| \geq 2\alpha$.

Second Case ($|c| = 1$). In this case we have $c = \{v\}$ for some vertex $v \in V$. Without loss of generality, let $w[c] = |w[c]| \geq 0$ – otherwise we can flip the sign of all weights again. In contrast to the previous case, a subset of $\bigcup M \cup c$ might induce edges in M and subedges of c. Those edges are induced by $V' = \bigcup M \setminus c$. For any $X' \subseteq V'$ we have $|w[X']| < \alpha$, otherwise we are done. Observe that $w[\bigcup M^+] = |M^+| + w[c] + w[\bigcup M^+]$. The statement follows immediately for $w[\bigcup M^+]| \geq \alpha$. Otherwise we obtain

$$\begin{aligned} |M^+| + w[c] &< \alpha + w[\bigcup M^+] < 2\alpha \quad \text{and} \\ |w[\bigcup M^-]| &= |M^-| - w[c] + w[\bigcup M^-] \\ &= |M| - |M^+| - w[c] - w[\bigcup M^-] > |M| - 3\alpha. \end{aligned}$$

The claim follows with $|M| \geq 4\alpha$.

Third Case ($|c| \geq 2$). Observe that in this case $\bigcup M$ may induce edges that are neither in $E(c)$ nor subedge of c, but that intersect c (and other vertices covered by M). This is because M contains only edges of $E(c)$. Let us recursively define the following subsets of $E\left[\bigcup M^\sigma\right]$ for $\sigma \in \{-,+\}$:

$$\begin{aligned} E^\sigma_\emptyset &= E\left[\left(\bigcup M^\sigma \setminus c\right)\right] \setminus E[c]; \\ E^\sigma_{c'} &= E\left[c' \cup \left(\bigcup M^\sigma \setminus c\right)\right] \setminus \bigcup_{f \subsetneq c'} E^\sigma_f \setminus E[c] \quad \text{for every } c' \subseteq c. \end{aligned}$$

For all $c' \subseteq c$ these sets are disjoint and every edge $e \in E[\bigcup M^\sigma]$ is either in one of these sets or in $E[c]$. To be precisely, an edge $e \in E[\bigcup M^\sigma]$ is in $E^\sigma_{e\cap c}$ if it is a proper superset of c. Otherwise $e \subseteq c$ (if $e \subseteq c$). Note that $M^\sigma = E^\sigma_c$. We will now show that there is a set $c' \subseteq c$ and such that the absolute value of the induced weight of $c' \cup (\bigcup M^- \setminus c)$ or $c' \cup (\bigcup M^+ \setminus c)$ is at least α. Assume this is not the case and that for all subsets $X \subseteq \bigcup M$ we have

$$|w[X]| < \alpha \qquad (1)$$

for any $c' \subseteq c$ and $\sigma \in \{-,+\}$. Observe that

$$E\left[c' \cup \left(\bigcup M^\sigma \setminus c\right)\right] = E^\sigma_{c'} \mathbin{\dot\cup} E[c'] \mathbin{\dot\cup} \bigcup_{f \subsetneq c'} E^\sigma_f. \tag{2}$$

Let $w^\sigma_{c'} = \sum_{e \in E^\sigma_{c'}} w(e)$ and note that this is the total weight of the edges in $E^\sigma_{c'}$. Hence:

$$w\left[c' \cup \left(\bigcup M^\sigma \setminus c\right)\right] = w^\sigma_{c'} + w[c'] + \sum_{f \subsetneq c'} w^\sigma_f. \tag{3}$$

For any $k \in \{0, \dots, |c|\}$ let $w_k = 2^{2^k - 1} \cdot 2\alpha$. We show that w_k is an upper bound in the sense that $|w^\sigma_{c'}| < w_k$ for $c' \subseteq c$ and $k = |c'|$. By (1) and (3) we deduce that

$$|w^\sigma_\emptyset| = \left| w\left[\left(\bigcup M^\sigma \setminus c\right)\right] - w[\emptyset] \right| < 2\alpha = w_0.$$

Otherwise either $|w[\emptyset]| \geq \alpha$ or $|w[(\bigcup M^\sigma \setminus c)]| \geq \alpha$. Observe that $2\alpha + (2^{k-1} - 1)w_{k-1} \leq w_k$ for $k > 0$. Then for $\emptyset \neq c' \subseteq c$ it follows by induction that

$$\begin{aligned} |w^\sigma_{c'}| &= \left| w\left[c' \cup \left(\bigcup M^\sigma \setminus c\right)\right] - w[c'] - \sum_{f \subsetneq c'} w^\sigma_f \right| \\ &< \alpha + \alpha + \sum_{k=0}^{|c'|-1} \binom{|c'|}{k} w_k \\ &\leq 2\alpha + (2^{|c'|} - 1) w_{|c'|-1} \\ &\leq w_{|c'|}. \end{aligned}$$

Since $|M| \geq 2\alpha \cdot 2^{2^d} \geq 4\alpha \cdot 2^{2^{|c|}-1} = 2 \cdot w_{|c|}$ either $|M^+| \geq w_{|c|}$ or $|M^-| \geq w_{|c|}$. Recall that $M^\sigma = E^\sigma_c$ and, thus, with $|w^\sigma_c| \geq |M^\sigma|$ we get that our assumption must be wrong. ◀

▶ **Corollary 22.** p_α-d-UNBALANCED-SUBGRAPH *has a kernel with* $O\big(\alpha^{2^d}\big)$ *vertices.*

Proof. If Rule 4 cannot be applied, there is no such subedge $c \subseteq e \in E$ that meets the two properties of the rule. However, this also implies that there is no subedge c that only meets the first property because otherwise there would be an inclusionwise largest superset (at last a size-$(d-1)$ edge), which does. Therefore, for any subedge $c \subseteq e \in E$ it holds that $|\operatorname{link}(c)| < (i^i 2\alpha \cdot 2^{2^d})^{2^i - 1}$ where $i = d - |c|$. Especially for $c = \emptyset$ we have $|\operatorname{link}(c)| < (2d^d 2^{2^d}\alpha)^{2^d - 1}$. By the definition of the link we have $|E| \leq |\operatorname{link}(\emptyset)| + 1$. Thus, the total number of edges is at most $(2d^d 2^{2^d}\alpha)^{2^d - 1}$. Hence, we get a kernel of size $O\big(\alpha^{2^d}\big)$. ◀

Proof of Theorem 3. Combine Corollary 18 with Corollary 22. ◀

If d is not a constant but a parameter, then, by an exhaustive application of the rules above, we obtain reduced instances of size $O\big(d(2d^d 2^{2^d}\alpha)^{2^d - 1}\big)$. Note that there could be up to n^d possible subedges, therefore it is not clear that we obtain a kernel. However, a closer look at Rule 4 reveals that we are not forced to consider these subedges – we can directly return a trivial yes-instance if $|E| \geq (2d^d 2^{2^d}\alpha)^{2^d - 1}$.

▶ **Corollary 23.** $\mathrm{p}_{\alpha,d}$-UNBALANCED-SUBGRAPH $\in$ FPT

Proof. A simple induction shows that when Rule 1 and Rule 2 were applied exhaustively and $|E| \geq (2d^d 2^{2^d}\alpha)^{2^d - 1}$, Rule 4 can be applied on $c = \emptyset$ because $E \setminus \{\emptyset\} = \operatorname{link}(\emptyset)$ or on some larger subedge. Hence we can return a trivial yes-instance whenever $|E| \geq (2d^d 2^{2^d}\alpha)^{2^d - 1}$. ◀

▶ **Corollary 24.** $\mathrm{p}_{\alpha,d}$-ABS-DNF $\in$ FPT

▶ **Corollary 25.** α-d-ABS-DNF $\in$ P

4.2 From Disjunctive Normal Form to Conjunctive Normal Form

Similar to Lemma 13 in Section 3, the goal of this section is to companion Corollary 24 with a CNF-version of the claim.

▶ **Lemma 26.** $\mathrm{p}_{\alpha,d}$-ABS-CNF $\leq_{fpt}$ $\mathrm{p}_{\alpha,d}$-ABS-DNF

Proof. Let (ϕ, w, α) be an instance of $\mathrm{p}_{\alpha,d}$-ABS-CNF and let $\Phi = \mathrm{clauses}(\phi)$. The idea of the reduction is to successively replace every disjunction in Φ by a set of conjunctions with new weights such that we have for the resulting pair (Φ', w'):

$$\sum_{\sigma\in\Phi,\beta\models\sigma} w(\sigma) = \sum_{\sigma\in\Phi',\beta\models\sigma} w'(\sigma).$$

Let c be some disjunction in Φ. Recall that c contains at most d variables. There is a set X of at most $2^{|\,\mathrm{vars}(c)|}$ conjunctions such that we have for every truth assignment $\beta\colon \mathrm{vars}(\phi) \to \{\mathit{true}, \mathit{false}\}$ (i) $\beta \models c$ iff β satisfies exactly one formula in X, and (ii) $\beta \not\models c$ iff β satisfies no formula in X. Such a set X can easily be computed using the truth table of c, which is possible in the reduction as d is a parameter. Let $w'(c') = w(c)$ for any $c' \in X$, and $w'(\tilde{c}) = w(\tilde{c})$ for any $\tilde{c} \in \Phi \setminus \{c\}$.

For the reduction, we replace Φ by $\Phi' = \Phi\setminus\{c\}\cup X$ and update w to w' accordingly as long as there is a disjunction $c \in \Phi$. Note that this will replace exactly $|\,\mathrm{clauses}(\phi)|$ disjunctions with at most 2^d conjunctions each, which implies that the reduction can be carried out by an fpt-algorithm. The resulting pair (Φ', w') satisfies for every truth assignment β:

$$\begin{aligned}
&\sum_{\sigma\in\Phi,\beta\models\sigma} w(\sigma)\\
&= \sum_{\sigma\in\Phi\setminus\{c\},\beta\models\sigma} w(\sigma) \quad + \quad \begin{cases} w(c) & \text{if } \beta \models c;\\ 0 & \text{otherwise;}\end{cases}\\
&= \sum_{\sigma\in\Phi\setminus\{c\},\beta\models\sigma} w(\sigma) \quad + \quad \sum_{\sigma\in X,\beta\models c} w(c)\\
&= \sum_{\sigma\in\Phi\setminus\{c\},\beta\models\sigma} w'(\sigma) \quad + \quad \sum_{\sigma\in X,\beta\models c} w'(\sigma)\\
&= \sum_{\sigma\in\Phi\setminus\{c\}\cup X,\beta\models\sigma} w'(\sigma).
\end{aligned}$$ ◀

▶ **Corollary 27.** $\mathrm{p}_{\alpha,d}$-ABS-CNF, $\mathrm{p}_{\alpha,d}$-ABS-MONOTONE-CNF $\in$ FPT

5 Application: Absolute Integer Optimization

We have seen that $\mathrm{p}_{\alpha,d}$-ABS-MONOTONE-DNF $\in$ FPT. We can generalize this problem to an algebraic optimization problem: Given a sum of products (a *multivariate polynomial* over binary variables), find an assignment such that the absolute value of the sum is at least α. Using binary variables, this problem is essentially the same problem as ABS-MONOTONE-DNF (in fact, optimization over DNFs is sometimes called sum of products). However, what happens if we do not have binary variables, but arbitrary integers from some given domain? We prove in this section that the problem remains tractable for similiar parameters.

Let us represent a multivariate polynomial $p_{A,w}(z_1, \ldots, z_n)$ over n variables $z_1, \ldots, z_n$ by an $n \times m$ matrix A and a vector w of length m as follows: Define $A\langle z, i, j\rangle$ to be $z^{A_{i,j}}$ if $z \neq 0$ or $A_{i,j} \neq 0$ and 1 otherwise; then write $p_{A,w}(z_1, \ldots, z_n) = \sum_{j=1}^{m} w_j \cdot \prod_{i=1}^{n} A\langle z_i, i, j\rangle$.

▶ **Problem 28** ($\mathrm{p}_{\alpha,d}$-ABS-IO).

Instance: *An $n \times m$ matrix $A \in \mathbb{N}_{\geq 0}^{n\times m}$, a weight vector $w \in \mathbb{Z}^m$, two bounding vectors $b_{min} \in (\mathbb{Z} \cup \{-\infty\})^n$ and $b_{max} \in (\mathbb{Z} \cup \{\infty\})^n$, and a target value $\alpha \in \mathbb{N}$.*
Parameter: α, $d = \max_{j\in\{1,\dots,m\}} \sum_{i=1}^{n} A_{i,j}$
Question: *Is there a solution $x = (x_1, x_2, \dots, x_n) \in \mathbb{Z}^n$ such that $|p_{A,w}(x)| \geq \alpha$ and for all $i \in \{1,\dots,n\}$ we have $(b_{min})_i \leq x_i \leq (b_{max})_i$?*

Here the parameter d is the maximal number of variables that do occur in any product of the polynomial. If there is no exponent larger than 1, then d is the number of variables in every product or *monomial.*

In this problem we restrict the domain of each variable to an interval. One might ask whether we could allow arbitrary linear inequations instead of upper and lower bounds, which is common in linear programming. However, in this case the problem becomes W[1]-hard:

▶ **Lemma 29.** $\mathrm{p}_{\alpha,d}$-ABS-IO *with linear inequation constraints is* W[1]*-hard.*

Proof. To prove this we turn an instance of p_k-INDEPENDENT-SET into an instance of $\mathrm{p}_{\alpha,d}$-ABS-IO with additional inequations. For a graph $G = (\{1,\dots,n\}, E)$ and a number $k \in \mathbb{N}$, let $A \in \mathbb{Z}^{n\times n}$ be the identity matrix, $w = (1,\dots,1) \in \mathbb{Z}^n$, and $\alpha = k$. Further set $b_{min} = (0)^n$, $b_{max} = (1)^n$, and let $x_u + x_v \leq 1$ for every edge $\{u, v\} \in E$ be additional constraints.

Since all weights in w are positive, the absolute value is the exact value of the sum. Every size-k independent has now a corresponding variable assignment and vice versa. ◀

Observe that variables may have a huge or possibly infinite domain of possible values and, hence, it is not obvious that $\mathrm{p}_{\alpha,d}$-ABS-IO is in FPT. However, we already mentioned that $\mathrm{p}_{\alpha,d}$-ABS-MONOTONE-DNF is equivalent to $\mathrm{p}_{\alpha,d}$-ABS-IO if (i) the domain of all variables is $\{0, 1\}$ (i. e., $b_{min} = (0)^n$ and $b_{max} = (1)^n$), and if (ii) we have no exponents (i. e., $A \in \{0, 1\}^{m\times n}$).

Therefore, we only have to consider the remaining cases. If (i) holds but (ii) does not, we can simply replace all nonzero-values in A by 1. This does not change the value of $p_{A,w}(x_1,\dots,x_n)$ as by (i) we have only binary variables and w remains the same.

If (i) does not hold, we use a set of reduction rules to transform an instance over an arbitrary domain into one in which the domain is a superset of $\{0, 1\}$. This is, of course, not the same as property (i). However, while adapting the domain, we can transform the instance such that it is reducible to an instance of $\mathrm{p}_{\alpha,d}$-UNBALANCED-SUBGRAPH with the following property: Rule 4 either identifies it as a yes-instance or does nothing. This property can be used by an algorithm for $\mathrm{p}_{\alpha,d}$-ABS-IO with the following trick: First modify the domain such that it contains $\{0, 1\}$; then virtually shrink the domain to $\{0, 1\}$ (this restricts the solution space and, thus, may turn a yes-instance to a no-instance, but may not turn a no-instance to a yes-instance); perform the reduction and apply Rule 4; either deduce that we are dealing with a yes-instance (trivial decision of Rule 4) or that the instance is small (as the rule did nothing). In the latter case restore the larger domain and explore a search tree to solve the problem (the size of the search tree is bounded, as Rule 4 did not trigger).

Let us first describe the reduction rules. As before, we assume that a rule may only be applied if rules with smaller numbers were applied exhaustively. The first rule removes unnecessary variables, products, and constraints; or detects that the instance has no solution.

▶ **Rule 5.** *Apply the following modifications if possible:*
1. *If $w_j = 0$ for some j, then remove the j-th column from A and w.*
2. *If there is some i such that $A_{i,j} = 0$ for every j, then remove the i-th row from A, $(b_{min})_i$, and $(b_{max})_i$.*

3. *If there is some i with $(b_{max})_i < (b_{min})_i$, then return a trivial no-instance.*
4. *If there is some i with $(b_{max})_i = (b_{min})_i$, then for every j where $A_{i,j} > 0$ replace w_j by $w_j \cdot (b_{max})_i^{A_{i,j}}$ and set $A_{i,j} = 0$.*
5. *If there are two equal columns j_0, j_1 in A, then replace w_{j_0} by $w_{j_0} + w_{j_1}$ and remove the j_1th column from A and w.*

▶ **Lemma 30.** *Rule 5 is safe.*

Proof. For Item 1 and 2 observe that neither variables that do not occur in the represented polynomial nor products that contain zero as a factor contribute to the value of the polynomial. Item 3 follows from the fact that there cannot be a solution if the domain of a variable is the empty set. Finally, Item 4 follows by the distributivity of addition and multiplication. ◀

If this rule cannot be applied, the domain of any variable contains at least two consecutive values. With the next rule we change and shift the domains of all variables such that the domain of every variable contains 0 and 1.

▶ **Rule 6.** *Apply the following modifications if possible:*
1. *If there is an i with $\{0,1\} \not\subseteq [(b_{min})_i, (b_{max})_i]$ and $(b_{min})_i \in \mathbb{Z}$, then for every column j_0 where $A_{i,j_0} > 0$ add $c := A_{i,j_0}$ new $j_1, j_2, \ldots, j_c$-th columns such that*

$$A_{i',j_k} = \begin{cases} A_{i',j_0} & \text{if } i' \neq i \\ A_{i,j_0} - k & \text{if } i' = i \end{cases} \quad \text{and} \quad w_{j_k} = \binom{c}{k} \cdot w_{j_0} \cdot (b_{min})_i^k \quad \text{for } k \in \{1, \ldots, c\}.$$

   Finally set $(b_{min})_i$ to 0 and $(b_{max})_i$ to $(b_{max})_i - (b_{min})_i$.
2. *If there is an i with $(b_{min})_i = -\infty$ and $(b_{max})_i \leq 0$, then replace w_j by $-w_j$ for every j where $A_{i,j}$ is odd, and set $(b_{min})_i$ to $-(b_{max})_i$ and $(b_{max})_i$ to ∞.*

▶ **Lemma 31.** *Rule 6 is safe.*

The first subrule shifts the domain of variables that have a finite lower bound. Variables without such a bound are turned into variables with a finite lower bound by the second subrule. The first subrule may then be applied again.

As soon as none of the rules can be applied, every variable that occurs in the polynomial $p_{A,w}$ has a domain that is a superset of $\{0,1\}$. We can turn our reduced instance into an instance of $\text{p}_{\alpha,d}$-ABS-DNF, which directly translates into an equivalent instance of $\text{p}_{\alpha,d}$-UNBALANCED-SUBGRAPH.

Thereby we might turn an ABS-IO yes-instance into a no-instance of UNBALANCED-SUBGRAPH. Recall that running the kernelization algorithm from Section 4 (especially Rule 4) either returns a trivial yes-instance or does nothing. After applying Rule 5 on the initial instance, rules 1 and 2 cannot be applied on the hypergraph. Therefore, the inital instance is a yes-instance or the size of the matrix is bounded by the parameters (because (H, w) is a kernel). Note that this does not imply that we obtain a kernel: The values in the bounding vectors are eventually not bounded yet. However, we show that such problematic instances can be reduced to a set of equivalent instances in which we have control over all these values.

▶ **Theorem 4.** $\text{p}_{\alpha,d}$-ABS-IO $\in$ FPT

Proof. Let us firstly present some branching rule for our problem:

▶ **Branching Rule 1.** *Let there be some i^* where $(b_{max})_{i^*} - (b_{min})_{i^*} \geq 2e\alpha$ with $e = \max_j A_{i^*,j}$. Let $w^{(k)} \in \mathbb{Z}^m$ with $w_j^{(k)} = w_j$ if $A_{i^*,j} = k$ and $w_j^{(k)} = 0$ otherwise, and let A' be the matrix obtained by setting all values in the i^*-th row of A to 0. Then $(A, w, b_{min}, b_{max}, \alpha) \in$ ABS-IO iff there is some $k \in \{0, \ldots, e\}$ such that $(A', w^{(k)}, b_{min}, b_{max}, \alpha_k) \in$ ABS-IO where $\alpha_k = 1$ for $k > 1$ and $\alpha_0 = \alpha$.*

▷ Claim 32. Branching Rule 1 is safe.

Proof. Let $(x_1, \ldots, x_n)$ be an arbitrary solution for the input instance. By term rewriting it follows that

$$\begin{aligned} p_{A,w}(x_1, \ldots, x_n) &= \sum_{j=1}^{m} w_j \cdot \prod_{i=1}^{n} A\langle x_i, i, j\rangle \\ &= \sum_{k=0}^{e} \left(\sum_{\substack{j\in\{1,\ldots,m\} \\ A_{i^*,j}=k}} w_j \cdot \prod_{i\in\{1,\ldots,n\}} A\langle x_i, i, j\rangle \right) \\ &= \sum_{k=0}^{e} \left(\sum_{\substack{j\in\{1,\ldots,m\} \\ A_{i^*,j}=k}} w_j^{(k)} \cdot \prod_{\substack{i\in\{1,\ldots,n\} \\ i\neq i^*}} A'\langle x_i, i, j\rangle \cdot A\langle x_{i^*}, i^*, j, \rangle \right) \\ &= \sum_{k=0}^{e} \left(p_{A',w^{(k)}}(x_1, \ldots, x_n) \cdot A\langle x_{i^*}, i^*, k, \rangle \right). \end{aligned}$$

We show that one of the branching instance has a solution, too. If $p_{A',w^{(k)}}(x_1, \ldots, x_n) > 0$ for some $k > 0$, we are done. Otherwise $p_{A',w^{(k)}}(x_1, \ldots, x_n) = 0$ holds for every $k > 0$. Then the equation above implies $p_{A',w^{(0)}}(x_1, \ldots, x_n) = p_{A,w}(x_1, \ldots, x_n)$. Since $|p_{A,w}(x_1, \ldots, x_n)| \geq \alpha$, we get that $|p_{A',w^{(0)}}(x_1, \ldots, x_n)| \geq \alpha$, which means $(A', w^{(0)}, b_{min}, b_{max}, \alpha) \in$ ABS-IO.

It remains to show that the input instance is a yes-instance if one of the branching instances is a yes-instance. Firstly, consider $(A', w^{(0)}, b_{min}, b_{max}, \alpha)$. Note, that $p_{A',w^{(0)}}(x_1, \ldots, x_n)$ does not depend on x_{i^*} because $w_j^{(0)} = 0$ for every product that does include x_{i_*}. For the same reason it also holds that $p_{A,w}(x_1, \ldots, x_{i^*-1}, 0, x_{i^*+1}, \ldots, x_n) = p_{A',w^{(0)}}(x_1, \ldots, x_n)$. Therefore, we can turn any solution of $(A', w^{(0)}, b_{min}, b_{max}, \alpha)$ into a solution for the input instance $(A, w, b_{min}, b_{max}, \alpha)$ by replacing x_i by 0.

Now assume that $(A', w^{(0)}, b_{min}, b_{max}), \alpha)$ is a no-instance and consider the largest number $k \in \{1, \ldots, e\}$ such that $(A', w^{(k)}, b_{min}, b_{max}, 1)$ is a yes-instance. Let $(y_1, \ldots, y_n)$ be one of its solutions. Define the polynomial $p(z) := p_{A,w}(y_1, \ldots, y_{i^*-1}, z, y_{i^*+1}, \ldots, y_n)$. It follows that $p(z) = \sum_{\ell=0}^{k} c_\ell \cdot z^\ell$ with $c_\ell = p_{A',w^{(\ell)}}(y_1, \ldots, y_n)$. Note that for every z we have $|p_{A',w^{(k)}}(y_1, \ldots, y_{i^*-1}, z, y_{i^*+1}, \ldots, y_n)| \geq 1$ as $A'_{i^*,j} = 0$ for every column j and, thus, $c_k \neq 0$. We will now argue by curve sketching over $\mathbb{R}$ that there is a $z \in \{(b_{min})_i, \ldots, (b_{max})_i\} \cap \mathbb{Z}$ with $|p(z)| \geq \alpha$. Any interval $[a, b]$ with $b - a > 2\alpha$ where either $(a, b]$ or $[a, b)$ contains no extremum of the polynomial contains at least one value $z \in \mathbb{Z}$ where $|p(z)| \geq \alpha$. This is because the polynomial is strictly increasing or decreasing on such an interval and the domain and image are restricted to $\mathbb{Z}$. The polynomial p has a degree of k, which is at most e, and, thus, it has at most $e - 1$ extreme points. Since $(b_{max})_i - (b_{min})_i \geq 2e\alpha$, there is such an interval within $[(b_{min})_i; (b_{max})_i]$. Hence, there is a $z \in [(b_{min})_i; (b_{max})_i]$ with $|p(z)| \geq \alpha$. By the definition of $p(z)$, we finally get that $(y_1, \ldots, y_{i-1}, z, y_{i+1} \ldots, y_n)$ is a solution for the input instance. ◁

Note that this rule reduces the number of variables on which the value of the polynomial depends. Recall Rule 5: As soon as a row contains only zeroes, it can be safely removed. Since the rules output at most $e \leq d$ branches, an exhaustive application of all the reduction rule results in a search tree of size $O(d^n)$. This search tree computes $O(d^n)$ instances in total, all of which have no variables with large domain (more than $2d\alpha$ possible values).

We partition the remaining part of the proof into three parts: First, we present the algorithm sketched in the main text in detail; second, we prove that this algorithm correctly solves $\mathrm{p}_{\alpha,d}$-ABS-IO; and third, we argue that the algorithm in fpt-time.

The Algorithm. Our algorithm has four phases. In the first phase it exhaustively applies the reduction rules 5 and 6 to obtain an instance $(A, w, b_{min}, b_{max}, \alpha)$. If Rule 5 returns a trivial no-instance, we reject the initial input. As soon as no reduction rule can be applied, it follows from Rule 5 that every variable in $p_{A,w}$ has a domain of size at least two. By Rule 6 every domain must be a superset of $\{0, 1\}$.

In the second phase we use the kernelization algorithm for $\mathrm{p}_{\alpha,d}$-UNBALANCED-SUBGRAPH: For the instance $(A, w, b_{min}, b_{max}, \alpha)$ obtained from the previous phase, let $H = (V, E)$ be a d-hypergraph and let $w\colon E \to \mathbb{Z}$ be a weight function with $V = \{1, \dots, n\}$, $E = \{e_1, \dots, e_m\}$, $e_j = \{i \mid A_{i,j} > 0\}$, and $w(e_j) = w_j$. We test whether $|E| \geq (2d^d 2^{2^d}\alpha)^{2^d-1}$ and accept if so.

Otherwise $m = |E| \leq (2d^d 2^{2^d}\alpha)^{2^d-1}$ and $n \leq d \cdot m$. Then we continue with the third phase and exhaustively apply Branching Rule 1 as well as the reduction rules (Rule 5 and Rule 6). This gives us a search tree that computes a set of instance on which neither a reduction rule nor the branching rule is applicable. The domain of all variables in all of these instances is bounded by the parameters and, hence, we can solve them via "brute-force."

Proof of Correctness. Consider the first phase of the algorithm where rules 5 and 6 are applied in the given order as long as possible. Since the rules are safe, it is correct to stop and reject if Rule 5 returns a trivial no-instance. Otherwise we obtain an instance that has a solution if, and only if, the initial input instance does.

Since the reduction rules have been applied exhaustively in the first phase, the domain for every variable is a superset of $\{0, 1\}$ afterwards. Therefore, any solution for the constructed UNBALANCED-SUBGRAPH instance gives us a valid solution. By Rule 5 there is no isolated vertex in V nor an edge $e \in E$ with $w(e) = 0$ in the constructed hypergraph. This means the only rules that modify the hypergraph (Rule 1 and Rule 2) are not applicable. Because of that, from the proof of Corollary 23, and from Lemma 21, the test for the size of the hypergraph correctly identifies trivial yes-instances or does nothing. We can derive a solution $(x_1, \dots, x_n) \in \mathbb{Z}^n$ with $|p_{A,w}(x_1, \dots, x_n)| \geq \alpha$ from a solution X for our hypergraph where $|w[X]| \geq \alpha$ as follows by setting $x_i = 1$ if $i \in X$, and by setting $x_i = 0$ otherwise. This is correct as the term that we maximize for the UNBALANCED-SUBGRAPH instance is the same as $|p_{A,w}(x_1, \dots, x_n)|$, and because the domain of every variable contains 0 and 1.

In the third phase, the algorithm reduces the instance to a set of instances in which the domain of every variable is bounded by the parameter. Variables with a too large domain will be eliminated by the branching rule and a following application of Rule 5 to remove the row containing only zeros. From the safeness of the branching rule it directly follows that there is one yes-instance if, and only if, one of the final branching instances is a yes-instance. Since the number of possible solutions for each instance is finite, a brute-force algorithm will decide for each of these instances correctly whether it is a yes-instance.

Runtime Analysis. We first argue that applying the reduction rules exhaustively can be done in time $f(\alpha, d) \cdot \mu^c$, were μ is the encoding length of the input, f some computable function, and $c \in \mathbb{N}$ a constant.

Rule 6 removes a variable with a domain that is not a superset of $\{0, 1\}$ and no rule introduces new variables. Hence, Rule 6 is applied at most $O(n)$ times. Rule 5 can only be applied $O(n + m)$ times because subrules 1–4 can be applied only once initially; the fifth subrule can be applied after Rule 6. Hence the rules can only be applied a polynomial number times. Observe that almost all reduction rules are computable in polynomial time with respect to the size of the input instance. The sole exceptions are Rule 6.1 and Rule 5.4, which still run in fpt-time.

We also have to show that the size of subproblems does not exceed the algorithm's time bound of $f(\alpha, d) \cdot \mu^c$. The critical part is Rule 6.1 – in all other cases the instance becomes only smaller. Each time we apply the rule for some i, the number of columns j for which $A_{i,j} > 0$ is increased by a factor of at most d as $A_{i,j} \leq d$. Every new column arises directly or indirectly from one column that was in the initial instance and, hence, the size of the instance increases at most by a factor of d^d in total.

For the search tree note that once the kernelization was used in the second phase, n and m are bounded by the parameters. The height and size of the search tree is therefore bounded, too. Note that the kernelization itself runs in polynomial time by the definition of a kernelization algorithm. Finally, whenever Branching Rule 1 cannot be applied, we have to solve the instance at the leaves of the search tree. Since neither a reduction rule nor the branching rule are applicable, it holds that n, m, all values in the matrix, and all values in the bounding vectors are bounded by the parameters for these instances. Thus, the brute-force algorithm's runtime is bounded by some computable function that depends only on the parameters (note that the weights are not necessarily bounded, but this is no problem for a combinatorial algorithm that enumerates the solutions). ◀

6 Conclusion and Outlook

We considered several variants of the maximum satisfiability problem and showed that, as soon as we allow negative weights, those variants become W[1]-hard parameterized by the solution size α – even for monotone clauses of fixed size d. On the other hand, we obtained fixed-parameter tractability results parameterized by $\alpha + d$ if we optimize the absolute value of the target function. The latter result was obtained via a kernelization for the auxiliary hypergraph problem $\mathrm{p}_{\alpha,d}$-UNBALANCED-SUBGRAPHS, which we think may be of independent interest. Using these techniques, we were able to almost completely resolve the complexity of ABS-DNF, see Table 1 in the introduction for an overview. The only remaining case is α-ABS-DNF, i. e., the version in which the sought solution size α is constant while the size of the clauses is unbounded. We do not see how our techniques can be used for this version, as our algorithms for the unbalanced subgraph problem rely on hyperedges of bounded size.

Using a collection of additional reduction rules, we were able to generalize the results from $\mathrm{p}_{\alpha,d}$-ABS-DNF to $\mathrm{p}_{\alpha,d}$-ABS-IO, which tries to optimize the absolute value of the target function of a restricted integer optimization problem.

An interesting line of further research could be to study the *minimization* version of the problems presented within this paper. Usually, minimization and maximization problems have similar complexity, as one can perform some easy modifications such as multiplying all weights with -1. This is, however, not the case if we optimize the absolute value of the target function, as the following observation illustrates: Let d-MIN-ABS-MONOTONE-DNF be defined as d-ABS-MONOTONE-DNF, but with $\geq \alpha$ being replaced by $\leq \alpha$, then:

▶ **Observation 33.** INDEPENDENT-SET $\leq$ d-MIN-ABS-MONOTONE-DNF

Sketch of Proof. We use the same reduction as in Lemma 5 to reduce INDEPENDENT-SET to d-MAX-MONOTONE-DNF. Hence, we obtain weighted formula (ϕ, w) and seek an assignment of weight at least $\geq \alpha$ (without absolute values). Note that if such an assignment exists, then there is also one with weight $= \alpha$ as the independent set problem is monotone. We continue by adding the empty clause (which is always true in a DNF) and set its weight to $-\alpha$. Finally, we seek an solution with absolute value ≤ 0. ◀

Note that this reduction works for constant d and produces an instance with $\alpha = 0$, i. e., a parameterization by α and d alone is not enough. However, restricting, for instance, the weights may yield tractable subproblems that should be explored further.

References

1 N. Alon, G.Z. Gutin, E.J. Kim, S. Szeider, and A. Yeo. Solving max-r-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011.

2 A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability, Second Edition*, volume 185 of *Front. Artif. Intell. Appl.* IOS Press, 2021.

3 J. Chen, C. Xu, and J. Wang. Dealing with 4-variables by resolution: An improved maxsat algorithm. *Theor. Comput. Sci.*, 670:33–44, 2017.

4 B. Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

5 Y. Crama, O. Ekin, and P.L. Hammer. Variable and term removal from boolean formulae. *Discret. Appl. Math.*, 75(3):217–230, 1997.

6 R. Crowston, M. Fellows, G. Gutin, M. Jones, F. Rosamond, S. Thomassé, and A. Yeo. Simultaneously Satisfying Linear Equations Over F_2: MaxLin2 and Max-r-Lin2 Parameterized Above Average. In *FSTTCS*, volume 13, pages 229–240, 2011.

7 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

8 H. Dell, E.J. Kim, M. Lampis, V. Mitsou, and T.Mömke. Complexity and approximability of parameterized max-csps. *Algorithmica*, 79(1):230–250, 2017.

9 E. Eiben, R. Ganian, D. Knop, and S. Ordyniak. Unary integer linear programming with structural restrictions. In *IJCAI*, pages 1284–1290. ijcai.org, 2018.

10 E. Eiben, R. Ganian, D. Knop, and S. Ordyniak. Solving integer quadratic programming via explicit and structural restrictions. In *AAAI*, pages 1477–1484. AAAI Press, 2019.

11 F. Eisenbrand, C. Hunkenschröder, K.M. Klein, M. Koutecký, A. Levin, and S. Onn. An algorithmic theory of integer programming, 2019. `arXiv:1904.01361`.

12 P. Erdős and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 1(1):85–90, 1960.

13 J. Flum and M. Grohe. *Parameterized Complexity Theory*. EATCS. Springer, 2006.

14 R. Ganian and S. Ordyniak. Solving integer linear programs by exploiting variable-constraint interactions: A survey. *Algorithms*, 12(12):248, 2019.

15 R. Ganian and S. Szeider. New width parameters for model counting. In *SAT*, volume 10491 of *Lecture Notes in Computer Science*, pages 38–52. Springer, 2017.

16 M.R. Garey, D.S. Johnson, and L.J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

17 S. Gaspers and S. Szeider. Kernels for global constraints. In *IJCAI*, pages 540–545. IJCAI/AAAI, 2011.

18 S. Gaspers and S. Szeider. Guarantees and limits of preprocessing in constraint satisfaction and reasoning. *Artif. Intell.*, 216:1–19, 2014.

19 T. Gavenčiak, D. Knop, and M. Koutecký. Integer Programming in Parameterized Complexity: Three Miniatures. In *IPEC*, volume 115, pages 21:1–21:16, 2019.

20 T. Gavenčiak, M. Koutecký, and D. Knop. Integer programming in parameterized complexity: Five miniatures. *Accepted for Discrete Optimization*, 2020. doi:10.1016/j.disopt.2020.100596.

21 M. Grohe. The structure of tractable constraint satisfaction problems. In *MFCS*, volume 4162, pages 58–72. Springer, 2006.

22 K. Jansen, A. Lassota, and L. Rohwedder. Near-linear time algorithm for n-fold ilps via color coding. *SIAM J. Discret. Math.*, 34(4):2282–2299, 2020.

23 S. Khanna, M. Sudan, and D.P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *STOC*, pages 11–20, 1997.

24 M. Koutecký, A. Levin, and S. Onn. A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs. In *ICALP)*, volume 107 of *LIPIcs*, pages 85:1–85:14, 2018.

25 H.W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

26 M. Mahajan and V. Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *J. Algorithms*, 31(2):335–354, 1999.

27 M. Mahajan, V. Raman, and S. Sikdar. Parameterizing above or below guaranteed values. *J. Comput. Syst. Sci.*, 75(2):137–153, 2009.

28 N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to horn and binary clauses. In *SAT*, 2004.

29 M. Samer and S. Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

30 M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci.*, 76(2):103–114, 2010.

31 T.J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978.

32 S. Szeider. The parameterized complexity of k-flip local search for SAT and MAX SAT. *Discret. Optim.*, 8(1):139–145, 2011.