Best-Case and Worst-Case Sparsifiability of Boolean CSPs

Hubie Chen
Birkbeck, University of London, Malet Street, Bloomsbury, London WC1E 7HX, United Kingdom
hubie@dcs.bbk.ac.uk

Bart M. P. Jansen[1]
Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
b.m.p.jansen@tue.nl

Astrid Pieterse[2]
Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.pieterse@tue.nl

Abstract

We continue the investigation of polynomial-time sparsification for NP-complete Boolean Constraint Satisfaction Problems (CSPs). The goal in sparsification is to reduce the number of constraints in a problem instance without changing the answer, such that a bound on the number of resulting constraints can be given in terms of the number of variables n. We investigate how the worst-case sparsification size depends on the types of constraints allowed in the problem formulation (the constraint language). Two algorithmic results are presented. The first result essentially shows that for any arity k, the only constraint type for which no nontrivial sparsification is possible has exactly one falsifying assignment, and corresponds to logical OR (up to negations). Our second result concerns linear sparsification, that is, a reduction to an equivalent instance with $\mathcal{O}(n)$ constraints. Using linear algebra over rings of integers modulo prime powers, we give an elegant necessary and sufficient condition for a constraint type to be captured by a degree-1 polynomial over such a ring, which yields linear sparsifications. The combination of these algorithmic results allows us to prove two characterizations that capture the optimal sparsification sizes for a range of Boolean CSPs. For NP-complete Boolean CSPs whose constraints are *symmetric* (the satisfaction depends only on the number of 1 values in the assignment, not on their positions), we give a complete characterization of which constraint languages allow for a linear sparsification. For Boolean CSPs in which every constraint has arity at most three, we characterize the optimal size of sparsifications in terms of the largest OR that can be expressed by the constraint language.

2012 ACM Subject Classification Computing methodologies → Symbolic and algebraic algorithms, Theory of computation → Problems, reductions and completeness, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases constraint satisfaction problems, kernelization, sparsification, lower bounds

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.15

Related Version A full version is available at [4], `https://arxiv.org/abs/1809.06171v1`.

Acknowledgements We would like to thank Emil Jeřábek for the proof of Lemma 4.2.

[1] Supported by NWO Gravitation grant "Networks".
[2] Supported by NWO Gravitation grant "Networks".

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 15; pp. 15:1–15:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Background

The framework of constraint satisfaction problems (CSPs) provides a unified way to study the computational complexity of a wide variety of combinatorial problems such as CNF-Satisfiability, Graph Coloring, and Not-All-Equal SAT. The framework uncovers algorithmic approaches that simultaneously apply to several problems, and also identifies common sources of intractability. For the purposes of this discussion, a CSP is specified using a (finite) *constraint language*, which is a set of (finite) relations; the problem is to decide the satisfiability of a set of constraints, where each constraint has a relation coming from the constraint language. The fact that many problems can be viewed as CSPs motivates the following investigation: how does the complexity of a CSP depend its constraint language? A key result in this area is Schaefer's dichotomy theorem [18], which classifies each CSP over the Boolean domain as polynomial-time solvable or NP-complete.

Continuing a recent line of investigation [10, 12, 15], we aim to understand for which NP-complete CSPs an instance can be *sparsified* in polynomial time, without changing the answer. In particular, we investigate the following questions. Can the number of constraints be reduced to a small function of the number of variables n? How does the sparsifiability of a CSP depend on its constraint language? We utilize the framework of kernelization [5, 8, 16], originating in parameterized complexity theory, to answer such questions.

The first results concerning polynomial-time sparsification in terms of the number n of variables or vertices were mainly negative. Under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$ (which we tacitly assume throughout this introduction), Dell and van Melkebeek [7] proved a strong lower bound: For any integer $d \geq 3$ and positive real ε, there cannot be a polynomial-time algorithm that compresses any instance φ of d-CNF-SAT on n variables, into an equivalent SAT instance φ' of bitsize $\mathcal{O}(n^{d-\varepsilon})$. In fact, there cannot even be an algorithm that transforms such φ into small equivalent instances ψ of an *arbitrary* decision problem. Since an instance of d-CNF-SAT has at most $2^d n^d \in \mathcal{O}(n^d)$ distinct clauses, it can *trivially* be sparsified to $\mathcal{O}(n^d)$ clauses by removing duplicates, and can be compressed to size $\mathcal{O}(n^d)$ by storing it as a bitstring indicating for each possible clause whether or not it is present. The cited lower bound therefore shows that the trivial sparsification for d-CNF-SAT cannot be significantly improved; we say that the problem does not admit nontrivial (polynomial-time) sparsification. Following these lower bounds for SAT, a number of other results were published [6, 9, 14] proving other problems do not admit nontrivial sparsification either.

This pessimistic state of affairs concerning nontrivial sparsification algorithms changed several years ago, when a subset of the authors [12] showed that the d-Not-All-Equal SAT problem *does* have a nontrivial sparsification. In this problem, clauses have size at most d and are satisfied if the literals do not all evaluate to the same value. While there can be $\Omega(n^d)$ different clauses in an instance, there is an efficient algorithm that finds a subset of $\mathcal{O}(n^{d-1})$ clauses that preserves the answer, resulting in a compression of bitsize $\mathcal{O}(n^{d-1} \log n)$. The first proof of this result was based on an ad-hoc application of a theorem of Lovász [17]. Later, the underlying proof technique was extracted and applied to a wider range of problems [10]. This led to the following understanding: if each relation in the constraint language can be represented by a polynomial of degree at most d, in a certain technical sense, then this allows the number of constraints in an n-variable instance of such a CSP to be reduced to $\mathcal{O}(n^d)$. The sparsification for d-Not-All-Equal SAT is then explained by noting that such constraints can be captured by polynomials of degree $d-1$. It is therefore apparent that finding a low-degree polynomial to capture the constraints of a CSP is a powerful tool

to obtain sparsification algorithms for it. Finding such polynomials of a certain degree d, or determining that they do not exist, proved a challenging and time-intensive task (cf. [11]).

The polynomial-based framework [10] also resulted in some *linear* sparsifications. Since "1-in-d" constraints (to satisfy a clause, *exactly one* out of its $\leq d$ literals should evaluate to true) can be captured by *linear* polynomials, the 1-IN-d-SAT problem has a sparsification with $\mathcal{O}(n)$ constraints for each constant d. This prompted a detailed investigation into linear sparsifications for CSPs by Lagerkvist and Wahlström [15], who used the toolkit of universal algebra in an attempt to obtain a characterization of the Boolean CSPs with a linear sparsification. Their results give a necessary and sufficient condition on the constraint language of a CSP for having a so-called Maltsev embedding over an infinite domain. They also show that when a CSP has a Maltsev embedding over a *finite* domain, then this can be used to obtain a linear sparsification. Alas, it remains unclear whether Maltsev embeddings over infinite domains can be exploited algorithmically, and a characterization of the linearly-sparsifiable CSPs is currently not known.

Our contributions

We analyze and demonstrate the power of the polynomial-based framework for sparsifying CSPs using universal algebra, linear algebra over rings, and relational analysis. We present two new algorithmic results. These allow us to characterize the sparsifiability of Boolean CSPs in two settings, wherein we show that the polynomial-based framework yields *optimal* sparsifications. In comparison to previous work [10], our results are much more fine-grained and based on a deeper understanding of the reasons why a certain CSP *cannot* be captured by low-degree polynomials.

Algorithmic results. Our first result (Section 3) shows that, contrary to the pessimistic picture that arose during the initial investigation of sparsifiability, the phenomenon of nontrivial sparsification is widespread and occurs for almost all Boolean CSPs! We prove that if Γ is a constraint language whose largest constraint has arity k, then the *only* reason that CSP(Γ) does not have a nontrivial sparsification, is that it contains an arity-k relation that is essentially the k-ary OR (up to negating variables). When $R \subseteq \{0,1\}^k$ is a relation with $|\{0,1\}^k \setminus R| \neq 1$ (the number of assignments that fail to satisfy the constraint is not equal to 1), then it can be captured by a polynomial of degree $k-1$. This yields a nontrivial sparsification compared to the $\Omega(n^k)$ distinct applications of this constraint that can be in such an instance.

Our second algorithmic result (Section 4) concerns the power of the polynomial-based framework for obtaining linear sparsifications. We give a necessary and sufficient condition for a relation to be captured by a degree-1 polynomial. Say that a Boolean relation $R \subseteq \{0,1\}^k$ is *balanced* if there is *no* sequence of vectors $s_1, \ldots, s_{2n}, s_{2n+1} \in R$ for $n \geq 1$ such that $s_1 - s_2 + s_3 \ldots - s_{2n} + s_{2n+1} = u \in \{0,1\}^k \setminus R$. (The same vector may appear multiple times in this sum.) In other words: R is balanced if one cannot find an odd-length sequence of vectors in R for which alternating between adding and subtracting these vectors component-wise results in a 0/1-bitvector u that is outside R. For example, the binary OR relation 2-OR $= \{0,1\}^2 \setminus \{(0,0)\}$ is *not* balanced, since $(0,1) - (1,1) + (1,0) = (0,0) \notin$ 2-OR, but the 1-in-3 relation $R_{=1} = \{(1,0,0),(0,1,0),(0,0,1)\}$ is. We prove that if a Boolean relation R is balanced, then it can efficiently be captured by a degree-1 polynomial and the number of constraints that are applications of this relation can be reduced to $\mathcal{O}(n)$. Hence when *all* relations in a constraint language Γ are balanced—we call such a constraint language *balanced*—then CSP(Γ) has a sparsification with $\mathcal{O}(n)$ constraints. We also show

that, on the other hand, if a Boolean relation R is not balanced, then there does not exist a degree-1 polynomial over any ring that captures R in the sense required for application of the polynomial framework. The property of being balanced is (as defined) a universal-algebraic property; these results thus tightly bridge universal algebra and the polynomial framework.

Characterizations. The property of being balanced gives an easy way to prove that certain Boolean CSPs admit linear sparsifications. But perhaps more importantly, this characterization constructively exhibits a certain *witness* when a relation can *not* be captured by a degree-1 polynomial, in the form of the alternating sum of satisfying assignments that yield an unsatisfying assignment. In several scenarios, we can turn this witness structure against degree-1 polynomials into a lower bound proving that the problem does not have a linear sparsification. As a consequence, we can prove two fine-grained characterizations of sparsification complexity.

Characterization of symmetric CSPs with a linear sparsification (Section 5) We say that a Boolean relation is *symmetric* if the satisfaction of a constraint only depends on the *number* of 1-values taken by the variables (the *weight* of the assignment), but does not depend on the *positions* where these values appear. For example, "1-in-k"-constraints are symmetric, just as "not-all-equal"-constraints, but the relation $R_{a\to b} = \{(0,0),(0,1),(1,1)\}$ corresponding to the truth value of $a \to b$ is not. We prove that if a symmetric Boolean relation R is not balanced, then it can implement (Definition 2.7) a binary OR using constants and negations but without having to introduce fresh variables. Building on this, we prove that if such an unbalanced symmetric relation R occurs in a constraint language Γ for which CSP(Γ) is NP-complete, then CSP(Γ) does *not* admit a sparsification of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$. Consequently, we obtain a characterization of the sparsification complexity of NP-complete Boolean CSPs whose constraint language consists of symmetric relations: there is a linear sparsification if and only if the constraint language is balanced. This yields linear sparsifications in several new scenarios that were not known before.

Characterization of sparsification complexity for CSPs of low arity (Section 6) By combining the linear sparsifications guaranteed by balanced constraint languages with the nontrivial sparsification when the largest-arity relations do not have exactly one falsifying assignment, we obtain an exact characterization of the optimal sparsification size for all Boolean CSPs where each relation has arity at most three. For a Boolean constraint language Γ consisting of relations of arity at most three, we characterize the sparsification complexity of Γ as an integer $k \in \{1,2,3\}$ that represents the largest OR that Γ can implement using constants and negations, but without introducing fresh variables. Then we prove that CSP(Γ) has a sparsification of size $\mathcal{O}(n^k)$, but no sparsification of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, giving matching upper and lower bounds. Hence for all Boolean CSPs with constraints of arity at most three, the polynomial-based framework gives provably *optimal* sparsifications.

2 Preliminaries

For a positive integer n, define $[n] := \{1,2,\ldots,n\}$. For an integer q, we let $\mathbb{Z}/q\mathbb{Z}$ denote the integers modulo q. These form a field if q is prime, and a ring otherwise. We will use $x \equiv_q y$ to denote that x and y are congruent modulo q, and $x \not\equiv_q y$ to denote that they are incongruent modulo q. For statements marked with a star (★), the (full) proof can be found in the full version [4].

Parameterized complexity. A parameterized problem $\mathcal{Q}$ is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *generalized kernel for $\mathcal{Q}$ into $\mathcal{Q}'$ of size $h(k)$* is an algorithm that, on input $(x,k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance (x',k') such that: (i) $|x'|$ and k' are bounded by $h(k)$, and (ii) $(x',k') \in \mathcal{Q}'$ if and only if $(x,k) \in \mathcal{Q}$. The algorithm is a *kernel* for $\mathcal{Q}$ if $\mathcal{Q}' = \mathcal{Q}$.

Since a polynomial-time reduction to an equivalent sparse instance yields a generalized kernel, lower bounds against generalized kernels can be used to prove the non-existence of such sparsification algorithms. To relate the sparsifiability of different problems to each other, the following notion is useful.

▶ **Definition 2.1.** Let $\mathcal{P}, \mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *linear-parameter transformation* from $\mathcal{P}$ to $\mathcal{Q}$ is a polynomial-time algorithm that, given an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ of $\mathcal{P}$, outputs an instance $(x',k') \in \Sigma^* \times \mathbb{N}$ of $\mathcal{Q}$ such that the following holds:

1. $(x,k) \in \mathcal{Q}$ if and only if $(x',k') \in \mathcal{P}$, and
2. $k' \in \mathcal{O}(k)$.

It is well-known [1, 2] that the existence of a linear-parameter transformation from problem $\mathcal{P}$ to $\mathcal{Q}$ implies that any generalized kernelization lower bound for $\mathcal{P}$, also holds for $\mathcal{Q}$.

Operations, relations, and preservation. A *Boolean operation* is a mapping from $\{0,1\}^k$ to $\{0,1\}$, where k, a natural number, is said to be the *arity* of the operation; we assume throughout that operations have positive arity. From here, we define a *partial Boolean operation* in the usual way, that is, it is a mapping from a subset of $\{0,1\}^k$ to $\{0,1\}$. We say that a partial Boolean operation f of arity k is *idempotent* if $f(0,\ldots,0) = 0$ and $f(1,\ldots,1) = 1$; and, *self-dual* if for all $(a_1,\ldots,a_k) \in \{0,1\}^k$, when $f(a_1,\ldots,a_k)$ is defined, it holds that $f(\neg a_1,\ldots,\neg a_k)$ is defined and $f(a_1,\ldots,a_k) = \neg f(\neg a_1,\ldots,\neg a_k)$.

▶ **Definition 2.2.** A partial Boolean operation $f\colon \{0,1\}^k \to \{0,1\}$ is *balanced* if there exist integer values $\alpha_1,\ldots,\alpha_k$, called the *coefficients* of f, such that

- $\sum_{i\in[k]} \alpha_i = 1$,
- $(x_1,\ldots,x_k)$ is in the domain of f if and only if $\sum_{i\in[k]} \alpha_i x_i \in \{0,1\}$, and
- $f(x_1,\ldots,x_k) = \sum_{i\in[k]} \alpha_i x_i$ for all tuples in its domain.

A *relation* over the set D is a subset of D^k; here, k is a natural number called the *arity* of the relation. Throughout, we assume that each relation is over a finite set D. A *Boolean relation* is a relation over $\{0,1\}$.

▶ **Definition 2.3.** For each $k \geq 1$, we use k-OR to denote the relation $\{0,1\}^k \setminus \{(0,\ldots,0)\}$.

A *constraint language over D* is a finite set of relations over D; a *Boolean constraint language* is a constraint language over $\{0,1\}$. For a Boolean constraint language Γ, we define $\mathsf{CSP}(\Gamma)$ as follows.

$\mathsf{CSP}(\Gamma)$	**Parameter:** The number of variables $\lvert V\rvert$.
Input: A tuple $(\mathcal{C}, V)$, where $\mathcal{C}$ is a finite set of constraints, V is a finite set of variables, and each constraint is a pair $R(x_1,\ldots,x_k)$ for $R \in \Gamma$ and $x_1,\ldots,x_k \in V$.	
Question: Does there exist a *satisfying assignment*, that is, an assignment $f\colon V \to \{0,1\}$ such that for each constraint $R(x_1,\ldots,x_k) \in \mathcal{C}$ it holds that $(f(x_1),\ldots,f(x_k)) \in R$?	

Let $f\colon \{0,1\}^k \to \{0,1\}$ be a partial Boolean operation, and let $T \subseteq \{0,1\}^n$ be a Boolean relation. We say that T is *preserved* by f when, for any tuples $t^1 = (t^1_1,\ldots,t^1_n),\ldots,t^k =$

$(t_1^k, \ldots, t_n^k) \in T$, if all entries of the tuple $(f(t_1^1, \ldots, t_1^k), \ldots, f(t_n^1, \ldots, t_n^k))$ are defined, then this tuple is in T. We say that a Boolean constraint language Γ is *preserved* by f if each relation in Γ is preserved by f. We say that a Boolean relation is *balanced* if it is preserved by all balanced operations, and that a Boolean constraint language is *balanced* if each relation therein is balanced.

Define an *alternating operation* to be a balanced operation $f\colon \{0,1\}^k \to \{0,1\}$ such that k is odd and the coefficients alternate between $+1$ and -1, so that $\alpha_1 = +1$, $\alpha_2 = -1$, $\alpha_3 = +1$, …, $\alpha_k = +1$. We have the following.

▶ **Proposition 2.4 (★).** *A Boolean relation R is balanced if and only if for all odd $k \geq 1$, the relation R is preserved by the alternating operation of arity k.*

We will use the following straightforwardly verified fact tacitly, throughout.

▶ **Observation 2.5.** *Each balanced operation is idempotent and self-dual.*

For $b \in \{0,1\}$, let $u_b\colon \{0,1\} \to \{0,1\}$ be the unary operation defined by $u_b(0) = u_b(1) = b$; let $\mathsf{major}\colon \{0,1\}^3 \to \{0,1\}$ to be the operation defined by $\mathsf{major}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$; and, let $\mathsf{minor}\colon \{0,1\}^3 \to \{0,1\}$ to be the operation defined by $\mathsf{minor}(x,y,z) = x \oplus y \oplus z$, where $\oplus$ denotes exclusive OR. We say that a Boolean constraint language Γ is *tractable* if it is preserved by one of the six following operations: u_0, u_1, $\wedge$, $\vee$, minor, major; we say that Γ is *intractable* otherwise. It is known that, in terms of classical complexity, the problem $\mathsf{CSP}(\Gamma)$ is polynomial-time decidable when Γ is tractable, and that the problem $\mathsf{CSP}(\Gamma)$ is NP-complete when Γ is intractable (see [3] for a proof; in particular, refer there to the proof of Theorem 3.21).

Constraint Satisfaction and Definability.

▶ **Assumption 2.6.** *By default, we assume in the sequel that the operations, relations, and constraint languages under discussion are Boolean, and that said operations and relations are of positive arity. We nonetheless sometimes describe them as being Boolean, for emphasis.*

▶ **Definition 2.7.** Let us say that a Boolean relation T of arity m is *cone-definable* from a Boolean relation U of arity n if there exists a tuple $(y_1, \ldots, y_n)$ where:

- for each $j \in [n]$, it holds that y_j is an element of $\{0,1\} \cup \{x_1, \ldots, x_m\} \cup \{\neg x_1, \ldots, \neg x_m\}$;
- for each $i \in [m]$, there exists $j \in [n]$ such that $y_j \in \{x_i, \neg x_i\}$; and,
- for each $f\colon \{x_1, \ldots, x_m\} \to \{0,1\}$, it holds that $(f(x_1), \ldots, f(x_m)) \in T$ if and only if $(\hat{f}(y_1), \ldots, \hat{f}(y_n)) \in U$. Here, $\hat{f}$ denotes the natural extension of f where $\hat{f}(0) = 0$, $\hat{f}(1) = 1$, and $\hat{f}(\neg x_i) = \neg f(x_i)$.

(The prefix *cone* indicates the allowing of **co**nstants and **ne**gation.)

▶ **Example 2.8.** Let $R = \{(0,0),(0,1)\}$ and let $S = \{(0,1),(1,1)\}$. We have that R is cone-definable from S via the tuple $(\neg x_2, \neg x_1)$; also, S is cone-definable from R via the same tuple.

When Γ is a constraint language over D, we use Γ^* to denote the expansion of Γ where each element of D appears as a relation, that is, we define Γ^* as $\Gamma \cup \{\{(d)\} \mid d \in D\}$.

The following is a key property of cone-definability; it states that relations that are cone-definable from a constraint language Γ may be simulated by the constraint language, and thus used to prove hardness results for $\mathsf{CSP}(\Gamma)$.

▶ **Proposition 2.9 (★).** *Suppose that Γ is an intractable constraint language, and that Δ is a constraint language such that each relation in Δ is cone-definable from a relation in Γ. Then, there exists a linear-parameter transformation from $\mathsf{CSP}(\Gamma^* \cup \Delta)$ to $\mathsf{CSP}(\Gamma)$.*

3 Trivial versus non-trivial sparsification

It is well known that k-CNF-SAT allows no non-trivial sparsification, for each $k \geq 3$ [7]. This means that we cannot efficiently reduce the number of clauses in such a formula to $\mathcal{O}(n^{k-\varepsilon})$. The k-OR relation is special, in the sense that there is exactly one k-tuple that is not contained in the relation. We show in this section that when considering k-ary relations for which there is more than one k-tuple not contained in the relation, a non-trivial sparsification is always possible. In particular, the number of constraints of any input can efficiently be reduced to $\mathcal{O}(n^{k-1})$. Using Lemmas 3.4 and 3.6, we will completely classify the constraint languages that allow a non-trivial sparsification as follows.

▶ **Theorem 3.1 (★).** *Let Γ be an intractable (Boolean) constraint language. Let k be the maximum arity of any relation $R \in \Gamma$. The following dichotomy holds.*

- *If for all $R \in \Gamma$ it holds that $|R| \neq 2^k - 1$, then $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n^{k-1})$ constraints that can be stored in $\mathcal{O}(n^{k-1} \log n)$ bits.*
- *If there exists $R \in \Gamma$ with $|R| = 2^k - 1$, then $\mathsf{CSP}(\Gamma)$ has no generalized kernel of bitsize $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

To obtain the kernels given in this section, we will heavily rely on the following notion for representing constraints by polynomials.

▶ **Definition 3.2.** Let R be a k-ary Boolean relation. We say that a polynomial p_u over a ring E_u *captures* an unsatisfying assignment $u \in \{0,1\}^k \setminus R$ with respect to R, if the following two conditions hold over E_u.

$$p_u(x_1, \ldots, x_k) = 0 \text{ for all } (x_1, \ldots, x_k) \in R, \text{ and} \tag{1}$$

$$p_u(u_1, \ldots, u_k) \neq 0. \tag{2}$$

The following Theorem is a generalization of Theorem 16 in [13]. The main improvement is that we now allow the usage of different polynomials, over different rings, for each $u \notin R$. Previously, all polynomials had to be given over the same ring, and each constraint was captured by a single polynomial.

▶ **Theorem 3.3 (★).** *Let $R \subseteq \{0,1\}^k$ be a fixed k-ary relation, such that for every $u \in \{0,1\}^k \setminus R$ there exists a ring $E_u \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u \text{ is a prime power}\}$ and polynomial p_u over E_u of degree at most d that captures u with respect to R. Then there exists a polynomial-time algorithm that, given a set of constraints $\mathcal{C}$ over $\{R\}$ over n variables, outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^d)$, such that any Boolean assignment satisfies all constraints in $\mathcal{C}$ if and only if it satisfies all constraints in $\mathcal{C}'$.*

The next lemma states that any k-ary Boolean relation R with $|R| < 2^k - 1$ admits a non-trivial sparsification. To prove the lemma, we show that such relations can be represented by polynomials of degree at most $k-1$, such that the sparsification can be obtained using Theorem 3.3. Since relations with $|R| = 2^k$ have a sparsification of size $\mathcal{O}(1)$, as constraints over such relations are satisfied by any assignment, it will follow that k-ary relations with $|\{0,1\}^k \setminus R| \neq 1$ always allow a non-trivial sparsification.

▶ **Lemma 3.4 (★).** *Let R be a k-ary Boolean relation with $|R| < 2^k - 1$. Let $\mathcal{C}$ be a set of constraints over $\{R\}$, using n variables. Then there exists a polynomial-time algorithm that outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^{k-1})$, such that a Boolean assignment satisfies all constraints in $\mathcal{C}'$ if and only if it satisfies all constraints in $\mathcal{C}$.*

Proof sketch. We will show that for every $u \in \{0,1\}^k \setminus R$, there exists a degree-$(k-1)$ polynomial p_u over $\mathbb{Q}$ that captures u, such that the result follows from Theorem 3.3. We will prove the existence of such a polynomial by induction on k. For $k = 1$, the lemma statement implies that $R = \emptyset$. Thereby, for any $u \notin R$, we simply choose $p_u(x_1) := 1$. This polynomial satisfies the requirements, and has degree 0. Let $k > 1$ and let $u = (u_1, \ldots, u_k) \in \{0,1\}^k \setminus R$. Since $|R| < 2^k - 1$, we can choose $w = (w_1, \ldots, w_k)$ such that $w \in \{0,1\}^k \setminus R$ and $w \neq u$. We distinguish two cases, depending on whether u and w agree on some position.

Suppose $u_i \neq w_i$ for all i, and assume for concreteness that $u = (0, \ldots, 0)$ and $w = (1, \ldots, 1)$. Then the polynomial $p_u(x_1, \ldots, x_k) := \prod_{i=1}^{k-1}(i - \sum_{j=1}^{k} x_j)$ suffices: $p_u(0, \ldots, 0) = \prod_{j=1}^{k-1} j \neq 0$, while for any $(x_1, \ldots, x_k) \in R$, it holds that $\sum_{i=1}^{k} x_i \in [k-1]$ and thereby $p_u(x_1, \ldots, x_k) = 0$; the product has a 0-term. Other values of u and w are handled similarly.

Now suppose $u_i = w_i$ for some $i \in [k]$, and assume for concreteness that $u_1 = w_1 = 1$. Define $R' := \{(x_2, \ldots, x_k) \mid (1, x_2, \ldots, x_k) \in R\}$ and let $u' := (u_2, \ldots, u_k)$. Since $(u_2, \ldots, u_k)$ and $(w_2, \ldots, w_k)$ are distinct tuples not in R', by induction there is a polynomial $p_{u'}$ of degree $k-2$ that captures u' with respect to R'. Then the polynomial $p_u(x_1, \ldots, x_k) := x_1 \cdot p_{u'}(x_2, \ldots, x_k)$ has degree $k-1$ and captures u with respect to R. ◀

To show the other part of the dichotomy, we will need the following theorem.

▶ **Theorem 3.5 (★).** *Let Γ be an intractable (Boolean) constraint language, and let $k \geq 1$. If there exists $R \in \Gamma$ such that R cone-defines k-OR, then $\mathsf{CSP}(\Gamma)$ does not have a generalized kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

The next lemma formalizes the idea that any k-ary relation with $|\{0,1\}^k \setminus R| = 1$ is equivalent to k-OR, up to negation of variables. The proof of the dichotomy given in Theorem 3.1 will follow from Lemma 3.4, together with the next lemma and Theorem 3.5.

▶ **Lemma 3.6 (★).** *Let R be a k-ary relation with $|R| = 2^k - 1$. Then R cone-defines k-OR.*

4 From balanced operations to linear sparsification

The main result of this section is the following theorem, which we prove below.

▶ **Theorem 4.1.** *Let Γ be a balanced (Boolean) constraint language. Then $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that are a subset of the original constraints. The kernel can be stored using $\mathcal{O}(n \log n)$ bits.*

To prove the theorem, we will use two additional technical lemmas. To state them, we introduce some notions from linear algebra. Given a set $S = \{s_1, \ldots, s_n\}$ of k-ary vectors in $\mathbb{Z}^k$, we define $\mathrm{span}_{\mathbb{Z}}(S)$ as the set of all vectors y in $\mathbb{Z}^k$ for which there exist $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}$ such that $y = \sum_{i \in [n]} \alpha_i s_i$. Similarly, we define $\mathrm{span}_q(S)$ as the set of all k-ary vectors y over $\mathbb{Z}/q\mathbb{Z}$, such that there exist $\alpha_1, \ldots, \alpha_n$ such that $y \equiv_q \sum_{i \in [n]} \alpha_i s_i$. For an $m \times n$ matrix S, we use s_i for $i \in [m]$ to denote the i'th row of S.

▶ **Lemma 4.2 (★).** *Let S be an $m \times n$ integer matrix. Let $u \in \mathbb{Z}^n$ be a row vector. If $u \in span_q(\{s_1, \ldots, s_m\})$ for all prime powers q, then $u \in span_{\mathbb{Z}}(\{s_1, \ldots, s_m\})$.*

▶ **Lemma 4.3 (★).** *Let q be a prime power. Let A be an $m \times n$ matrix over $\mathbb{Z}/q\mathbb{Z}$. Suppose there exists no constant $c \not\equiv_q 0$ for which the system $Ax \equiv_q b$ has a solution, where $b := (0, \ldots, 0, c)^T$ is the vector with c on the last position and zeros in all other positions.*

Then $a_m \in span_q(\{a_1, \ldots, a_{m-1}\})$.

Using these tools from linear algebra, we now prove the main sparsification result.

Proof of Theorem 4.1. We show that for all relations R in the balanced constraint language Γ, for all $u \notin R$, there exists a linear polynomial p_u over a ring $E_u \in \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u \text{ is a prime power}\}$ that captures u with respect to R. By applying Theorem 3.3 once for each relation $R \in \Gamma$, to reduce the number of constraints involving R to $\mathcal{O}(n)$, we then reduce any n-variable instance of $\mathsf{CSP}(\Gamma)$ to an equivalent one on $|\Gamma| \cdot \mathcal{O}(n) \in \mathcal{O}(n)$ constraints.

Suppose for a contradiction that there exists $R \in \Gamma$ and $u \notin R$, such that no prime power q and polynomial p over $\mathbb{Z}/q\mathbb{Z}$ exist that satisfy conditions (1) and (2). We can view the process of finding such a linear polynomial, as solving a set of linear equations whose unknowns are the coefficients of the polynomial. We have a linear equation for each evaluation of the polynomial for which we want to enforce a certain value.

Let $R = \{r_1, \ldots, r_\ell\}$. By the non-existence of p and q, the system

$$\begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \ldots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \ldots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{\ell,1} & r_{\ell,2} & \ldots & r_{\ell,k} \\ 1 & u_1 & u_2 & \ldots & u_k \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \equiv_q \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

has no solution for any prime power q and $c \not\equiv_q 0$. Otherwise, it is easy to verify that q is the desired prime power and $p(x_1, \ldots, x_k) := \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i$ is the desired polynomial.

The fact that no solution exists, implies that $(1, u_1, \ldots, u_k)$ is in the span of the remaining rows of the matrix, by Lemma 4.3. But this implies that for any prime power q, there exist coefficients $\beta_1, \ldots, \beta_\ell$ over $\mathbb{Z}/q\mathbb{Z}$ such that $u \equiv_q \sum \beta_i r_i$. Furthermore, since the first column of the matrix is the all-ones column, we obtain that $\sum \beta_i \equiv_q 1$. By Lemma 4.2, it follows that there exist integer coefficients $\gamma_1, \ldots, \gamma_\ell$ such that $\sum \gamma_i = 1$ and furthermore $u = \sum \gamma_i r_i$. But it immediately follows that $R \in \Gamma$ is not preserved by the balanced operation given by $f(x_1, \ldots, x_\ell) := \sum \gamma_i x_i$, which contradicts the assumption that Γ is balanced. ◀

The kernelization result above is obtained by using the fact that when Γ is balanced, the constraints in $\mathsf{CSP}(\Gamma)$ can be replaced by linear polynomials. We show in the next theorem that this approach fails when Γ is not balanced.

▶ **Theorem 4.4 (★).** *Let R be a k-ary relation that is not balanced. Then there exists $u \in \{0,1\}^k \setminus R$ for which there exists no polynomial p_u over any ring E that captures u with respect to R.*

5 Characterization of symmetric CSPs with linear sparsification

In this section, we characterize the symmetric constraint languages Γ for which $\mathsf{CSP}(\Gamma)$ has a linear sparsification.

▶ **Definition 5.1.** We say a k-ary Boolean relation R is *symmetric*, if there exists $S \subseteq \{0, 1, \ldots, k\}$ such that a tuple $x = (x_1, \ldots, x_k)$ is in R if and only if $\text{weight}(x) \in S$. We call S the set of *satisfying weights* for R.

We will say that a constraint language Γ is symmetric, if it only contains symmetric relations. We will prove the following theorem at the end of this section.

▶ **Theorem 5.2.** *Let Γ be a finite Boolean symmetric intractable constraint language.*

- *If* Γ *is balanced, then* $\mathsf{CSP}(\Gamma)$ *has a kernel with* $\mathcal{O}(n)$ *constraints that can be stored in* $\mathcal{O}(n \log n)$ *bits.*
- *If* Γ *is not balanced, then* $\mathsf{CSP}(\Gamma)$ *does not have a generalized kernel of size* $\mathcal{O}(n^{2-\varepsilon})$ *for any* $\varepsilon > 0$*, unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

To show this, we use the following lemma.

▶ **Lemma 5.3 (★).** *Let* R *be a* k*-ary symmetric relation with satisfying weights* $S \subseteq \{0, 1, \ldots, k\}$*. Let* $U := \{0, 1, \ldots, k\} \setminus S$*. If there exist* $a, b, c \in S$ *and* $d \in U$ *such that* $a - b + c = d$*, then* R *cone-defines* 2-OR.

Proof sketch. We will demonstrate the result in the case that $b \leq a$, $b \leq c$, and $b \leq d$; the other cases are similar. We use the following tuple to express $x_1 \vee x_2$.

$$(\underbrace{\neg x_1, \ldots, \neg x_1}_{(a-b) \text{ copies}}, \underbrace{\neg x_2, \ldots, \neg x_2}_{(c-b) \text{ copies}}, \underbrace{1, \ldots, 1}_{b \text{ copies}}, \underbrace{0, \ \ldots \ , 0}_{(k-d) \text{ copies}}).$$

Let $f \colon \{x_1, x_2\} \to \{0, 1\}$, then $(\neg f(x_1), \ldots, \neg f(x_1), \neg f(x_2), \ldots, \neg f(x_2), 1, \ldots, 1, 0, \ldots, 0)$ has weight $d \notin S$ when $f(x_1) = f(x_2) = 0$. It is easy to verify that in all other cases, the weight is one of $a, b, c \in S$ and hence the tuple belongs to R. The other cases are similar. ◀

We now give the main lemma that is needed to prove Theorem 5.2. It shows that if a relation is symmetric and not balanced, it must cone-define 2-OR.

▶ **Lemma 5.4.** *Let* R *be a symmetric (Boolean) relation of arity* k*. If* R *is not balanced, then* R *cone-defines* 2-OR.

Proof. Let f be a balanced operation that does not preserve R. Since f has integer coefficients, it follows that there exist (not necessarily distinct) $r_1, \ldots, r_m \in R$, such that $r_1 - r_2 + r_3 - r_4 \cdots + r_m = u$ for some $u \in \{0, 1\}^k \setminus R$ and odd $m \geq 3$. Thereby, $\text{weight}(r_1) - \text{weight}(r_2) + \text{weight}(r_3) - \text{weight}(r_4) \cdots + \text{weight}(r_m) = \text{weight}(u)$. Let S be the set of satisfying weights for R and let $U := \{0, \ldots, k\} \setminus S$. Define $s_i := \text{weight}(r_i)$ for $i \in [m]$, and $t = \text{weight}(u)$, such that $s_1 - s_2 + s_3 - s_4 \ldots + s_m = t$, and furthermore $s_i \in S$ for all i, and $t \in U$. We show that there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, such that the result follows from Lemma 5.3. We do this by induction on the length of the alternating sum.

If $m = 3$, we have that $s_1 - s_2 + s_3 = t$ and define $a := s_1$, $b := s_2$, $c := s_3$, and $d := t$.

If $m > 3$, we will use the following claim.

▶ **Claim 5.5 (★).** *Let* $s_1, \ldots, s_m \in S$ *and* $t \in U$ *such that* $s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$*. There exist distinct* $i, j, \ell \in [m]$ *with* i, j *odd and* ℓ *even, such that* $s_i - s_\ell + s_j \in \{0, \ldots, k\}$.

Use Claim 5.5 to find i, j, ℓ such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$. We consider two options. If $s_i - s_\ell + s_j \in U$, then define $d := s_i - s_\ell + s_j$, $a := s_i$, $b := s_\ell$, and $c := s_j$ and we are done. The other option is that $s_i - s_\ell + s_j = s \in S$. Replacing $s_i - s_\ell + s_j$ by s in $s_1 - s_2 + s_3 - s_4 \cdots + s_m$ gives a shorter alternating sum with result t. We obtain a, b, c, and d by the induction hypothesis.

Thereby, we have obtained $a, b, c \in S$, $d \in U$ such that $a - b + c = d$. It now follows from Lemma 5.3 that R cone-defines 2-OR. ◀

Using the lemma above, we can now prove Theorem 5.2.

Proof of Theorem 5.2. If Γ is balanced, it follows from Theorem 4.1 that $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits. Note that the assumption that Γ is symmetric is not needed in this case.

If the symmetric constraint language Γ is not balanced, then Γ contains a symmetric relation R that is not balanced. It follows from Lemma 5.4 that R cone-defines the 2-OR relation. Thereby, we obtain from Theorem 3.5 that $\mathsf{CSP}(\Gamma)$ has no generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

6 Low-arity classification

In this section, we will give a full classification of the sparsifiability for constraint languages that consist only of low-arity relations. The next results will show that in this case, if the constraint language is not balanced, it can cone-define the 2-OR relation.

▶ **Observation 6.1.** *Each relation of arity* 1 *is balanced.*

▶ **Theorem 6.2 (★).** *A relation of arity* 2 *is balanced if and only if it is not cone-interdefinable with the* 2-OR *relation.*

▶ **Theorem 6.3 (★).** *Suppose that* $U \subseteq \{0,1\}^3$ *is an arity* 3 *Boolean relation that is not balanced. Then, the* 2-OR *relation is cone-definable from* U*.*

Combining the results in this section with the results in previous sections, allows us to give a full classification of the sparsifiability of constraint languages that only contain relations of arity at most three. Observe that any k-ary relation R such that $R \neq \emptyset$ and $\{0,1\}^k \setminus R \neq \emptyset$ cone-defines the 1-OR relation. Since we assume that Γ is intractable in the next theorem, it follows that k is always defined and $k \in \{1,2,3\}$.

▶ **Theorem 6.4.** *Let* Γ *be an intractable Boolean constraint language such that each relation therein has arity* ≤ 3*. Let* $k \in \mathbb{N}$ *be the largest value for which* k*-OR can be cone-defined from a relation in* Γ*. Then* $\mathsf{CSP}(\Gamma)$ *has a kernel with* $\mathcal{O}(n^k)$ *constraints that can be encoded in* $\mathcal{O}(n^k \log k)$ *bits, but for any* $\varepsilon > 0$ *there is no kernel of size* $\mathcal{O}(n^{k-\varepsilon})$*, unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*.*

Proof. To show that there is a kernel with $\mathcal{O}(n^k)$ constraints, we do a case distinction on k.

- ($k = 1$) If $k = 1$, there is no relation in Γ that cone-defines the 2-OR relation. It follows from Observation 6.1 and Theorems 6.2 and 6.3 that thereby, Γ is balanced. It now follows from Theorem 4.1 that $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.
- ($k = 2$) If $k = 2$, there is no relation $R \in \Gamma$ with $|R| = 2^3 - 1 = 7$, as otherwise by Lemma 3.6 such a relation R would cone-define 3-OR which is a contradiction. Thereby, it follows from Theorem 3.1 that $\mathsf{CSP}(\Gamma)$ has a sparsification with $\mathcal{O}(n^{3-1}) = \mathcal{O}(n^2)$ constraints that can be encoded in $\mathcal{O}(n^2 \log n)$ bits.
- ($k = 3$) Given an instance $(\mathcal{C}, V)$, it is easy to obtain a kernel of with $\mathcal{O}(n^3)$ constraints by simply removing duplicate constraints. This kernel can be stored in $\mathcal{O}(n^3)$ bits, by storing for each relation $R \in \Gamma$ and for each tuple $(x_1, x_2, x_3) \in V^3$ whether $R(x_1, x_2, x_3) \in \mathcal{C}$. Since $|\Gamma|$ is constant and there are $\mathcal{O}(n^3)$ such tuples, this results in using $\mathcal{O}(n^3)$ bits.

It remains to prove the lower bound. By definition, there exists $R \in \Gamma$ such that R cone-defines the k-OR relation. Thereby, the result follows immediately from Theorem 3.5. Thus, $\mathsf{CSP}(\Gamma)$ has no kernel of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

7 Conclusion

In this paper we analyzed the best-case and worst-case sparsifiability of $\mathsf{CSP}(\Gamma)$ for intractable finite Boolean constraint languages Γ. First of all, we characterized those Boolean CSPs for which a nontrivial sparsification is possible, based on the number of non-satisfying assignments. Then we presented our key structural contribution: the notion of balanced constraint languages. We have shown that $\mathsf{CSP}(\Gamma)$ allows a sparsification with $\mathcal{O}(n)$ constraints whenever Γ is balanced. The constructive proof of this statement can be transformed into an effective algorithm to find a series of low-degree polynomials to capture the constraints, which earlier had to be done by hand. By combining the resulting upper and lower bound framework, we fully classified the symmetric constraint languages for which $\mathsf{CSP}(\Gamma)$ allows a linear sparsification. Furthermore, we fully classified the sparsifiability of $\mathsf{CSP}(\Gamma)$ when Γ contains relations of arity at most three, based on the arity of the largest OR that can be cone-defined from Γ. It follows from results of Lagerkvist and Wahlström [15] that for constraint languages of arbitrary arity, the exponent of the best sparsification size does not always match the arity of the largest OR cone-definable from Γ. (This will be described in more detail in the upcoming journal version of this work.) Hence the type of characterization we presented is inherently limited to low-arity constraint languages. It may be possible to extend our characterization to languages of arity at most four, however.

The ultimate goal of this line of research is to fully classify the sparsifiability of $\mathsf{CSP}(\Gamma)$, depending on Γ. In particular, we would like to classify those Γ for which $\mathcal{O}(n)$ sparsifiability is possible. In this paper, we have shown that Γ being balanced is a sufficient condition to obtain a linear sparsification; it is tempting to conjecture that this condition is also necessary.

We conclude with a brief discussion on the relation between our polynomial-based framework for linear compression and the framework of Lagerkvist and Wahlström [15]. They used a different method for sparsification, based on embedding a Boolean constraint language Γ into a constraint language Γ' defined over a larger domain D, such that Γ' is preserved by a Maltsev operation. This latter condition ensures that $\mathrm{CSP}(\Gamma')$ is polynomial-time solvable, which allows $\mathrm{CSP}(\Gamma)$ to be sparsified to $\mathcal{O}(n)$ constraints when D is finite. It turns out that the Maltsev-based linear sparsification is more general than the polynomial-based linear sparsification presented here: all finite Boolean constraint languages Γ that are balanced, admit a Maltsev embedding over a finite domain (the direct sum of the rings $\mathbb{Z}/q_u\mathbb{Z}$ over which the capturing polynomials are defined) and can therefore be linearly sparsified using the algorithm of Lagerkvist and Wahlström. Despite the fact that our polynomial-based framework is not more general than the Maltsev-based approach, it has two distinct advantages. First of all, there is a straight-forward decision procedure to determine whether a constraint can be captured by degree-1 polynomials, which follows from the proof of Theorem 4.1. To the best of our knowledge, no decision procedure is known to determine whether a Boolean constraint language admits a Maltsev embedding over a finite domain. The second advantage of our method is that when the polynomial framework for linear compression does not apply, this is witnessed by a relation in Γ that is violated by a balanced operation. As we have shown, in several scenarios this violation can be used to construct a sparsification lower bound to give provably optimal bounds.

It would be interesting to determine whether the Maltsev-based framework for sparsification is strictly more general than the polynomial-based framework. We are not aware of any concrete Boolean constraint language Γ for which $\mathrm{CSP}(\Gamma)$ admits a Maltsev embedding over a finite domain, yet is not balanced.

References

1 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.

2 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.

3 Hubie Chen. A Rendezvous of Logic, Complexity, and Algebra. *ACM Computing Surveys*, 42(1), 2009. doi:10.1145/1189056.1189076.

4 Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Best-case and Worst-case Sparsifiability of Boolean CSPs. *CoRR*, abs/1809.06171v1, 2018. arXiv:1809.06171v1.

5 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proc. 23rd SODA*, pages 68–81, 2012. doi:10.1137/1.9781611973099.6.

7 Holger Dell and Dieter van Melkebeek. Satisfiability Allows No Nontrivial Sparsification unless the Polynomial-Time Hierarchy Collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.

8 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

9 Bart M. P. Jansen. On Sparsification for Computing Treewidth. *Algorithmica*, 71(3):605–635, 2015. doi:10.1007/s00453-014-9924-2.

10 Bart M. P. Jansen and Astrid Pieterse. Optimal Sparsification for Some Binary CSPs Using Low-Degree Polynomials. In *Proc. 41st MFCS*, pages 71:1–71:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.71.

11 Bart M. P. Jansen and Astrid Pieterse. Optimal Data Reduction for Graph Coloring Using Low-Degree Polynomials. In *Proc. 12th IPEC*, pages 22:1–22:12, 2017. doi:10.4230/LIPIcs.IPEC.2017.22.

12 Bart M. P. Jansen and Astrid Pieterse. Sparsification Upper and Lower Bounds for Graph Problems and Not-All-Equal SAT. *Algorithmica*, 79(1):3–28, 2017. doi:10.1007/s00453-016-0189-9.

13 Bart M. P. Jansen and Astrid Pieterse. Optimal Sparsification for Some Binary CSPs Using Low-Degree Polynomials. *CoRR*, abs/1606.03233v2, 2018. arXiv:1606.03233v2.

14 Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point Line Cover: The Easy Kernel is Essentially Tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016. doi:10.1145/2832912.

15 Victor Lagerkvist and Magnus Wahlström. Kernelization of Constraint Satisfaction Problems: A Study Through Universal Algebra. In *Proc. 23rd CP*, pages 157–171, 2017. doi:10.1007/978-3-319-66158-2_11.

16 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a Guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161, 2012. doi:10.1007/978-3-642-30891-8_10.

17 Lásló Lovász. Chromatic number of hypergraphs and linear algebra. In *Studia Scientiarum Mathematicarum Hungarica 11*, pages 113–114, 1976.

18 Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proc. 10th STOC*, pages 216–226, 1978. doi:10.1145/800133.804350.