


Optimal Polynomial-Time Compression for Boolean Max CSP

Bart M. P. Jansen 

Eindhoven University of Technology, The Netherlands
b.m.p.jansen@tue.nl

Michał Włodarczyk 

Eindhoven University of Technology, The Netherlands
m.wlodarczyk@tue.nl

Abstract

In the Boolean maximum constraint satisfaction problem – MAX CSP(Γ) – one is given a collection of weighted applications of constraints from a finite constraint language Γ , over a common set of variables, and the goal is to assign Boolean values to the variables so that the total weight of satisfied constraints is maximized. There exists a concise dichotomy theorem providing a criterion on Γ for the problem to be polynomial-time solvable and stating that otherwise it becomes NP-hard. We study the NP-hard cases through the lens of kernelization and provide a complete characterization of MAX CSP(Γ) with respect to the optimal compression size. Namely, we prove that MAX CSP(Γ) parameterized by the number of variables n is either polynomial-time solvable, or there exists an integer $d \geq 2$ depending on Γ , such that:

1. An instance of MAX CSP(Γ) can be compressed into an equivalent instance with $\mathcal{O}(n^d \log n)$ bits in polynomial time,
2. MAX CSP(Γ) does not admit such a compression to $\mathcal{O}(n^{d-\epsilon})$ bits unless $\text{NP} \subseteq \text{co-NP/poly}$.

Our reductions are based on interpreting constraints as multilinear polynomials combined with the framework of constraint implementations. As another application of our reductions, we reveal tight connections between optimal running times for solving MAX CSP(Γ). More precisely, we show that obtaining a running time of the form $\mathcal{O}(2^{(1-\epsilon)n})$ for particular classes of MAX CSPs is as hard as breaching this barrier for MAX d -SAT for some d .

2012 ACM Subject Classification Theory of computation \rightarrow Fixed parameter tractability

Keywords and phrases constraint satisfaction problem, kernelization, exponential time algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.63

Related Version A full version of the paper is available at <https://arxiv.org/abs/2002.03443>.

Funding This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch).

Michał Włodarczyk: The author was supported by the Foundation for Polish Science (FNP).



1 Introduction

Background and motivation. The framework of constraint satisfaction problems (CSPs) allows the computational complexity of a large class of problems to be studied through a common lens [11]. A typical instance of such a problem asks whether it is possible to assign each of the variables x_1, \dots, x_n a value from a finite domain D , such that a given list of constraint applications is satisfied. A constraint is applied to a fixed number of variables, and indicates which combinations of values are legal. In the MAX CSP problem, the goal is to maximize the number of satisfied constraints. See Section 2 for formal definitions.



© Bart M. P. Jansen and Michał Włodarczyk;
licensed under Creative Commons License CC-BY
28th Annual European Symposium on Algorithms (ESA 2020).

Editors: Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders; Article No. 63; pp. 63:1–63:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The investigation of CSPs has led to deep theorems characterizing the complexity of a CSP based on the type of constraints allowed in the instance [7, 24]. For example, the long-awaited CSP dichotomy theorem [6, 35] provides a criterion separating the NP-complete from the polynomial-time solvable CSPs; the work of Khanna, Sudan, Trevisan, and Williamson characterizes how well the maximization version of a Boolean CSP can be approximated [21] (see [13, 20] for larger domains; see [12, 27] for optimal approximation factors); and Cai and Chen [8] present a dichotomy that separates CSPs for which the number of complex-weighted solutions can be counted in polynomial time, from those where the problem is #P-hard.

In this work we analyze the complexity of constraint satisfaction in an algorithmic regime that is currently far from understood: polynomial-time compression and kernelization [15]. Here, the goal is to analyze how much (in terms of the number of variables n) an instance can be compressed by a polynomial-time algorithm without changing the answer, and to understand how the compressibility depends on the type of available constraints. A *compression* is a polynomial-time algorithm that reduces instances of one problem to equivalent, small instances of a potentially different problem; a *kernelization* compresses to an instance of the same problem (see Section 2.4). A kernelization of small size allows an instance to be stored, manipulated, and solved more efficiently. It is therefore of interest to find the smallest possible kernelizations. Since every kernelization yields a compression, one can prove lower bounds on the size of kernelizations by establishing lower bounds on compressions.

In recent years, there have been a number of advances in the understanding of compressibility of CSPs [9, 14, 18, 25]. A foundational result by Dell and van Melkebeek [14] states that for $d \geq 3$, CNF-SAT with clauses of size at most d (d -CNF-SAT) parameterized by the number of variables n admits no (polynomial-time) compression of size $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{co-NP/poly}$ (which is known to imply a collapse of the polynomial hierarchy [34]). As an instance of d -CNF-SAT can trivially be compressed to $\mathcal{O}(n^d)$ bits via a bitstring that encodes for each of the $\mathcal{O}(n^d)$ possible clauses whether or not it is present in the instance, the d -CNF-SAT problem does not admit any non-trivial compression. The situation is different for the related problem d -NOT-ALL-EQUAL SAT (d -NAE-SAT), which is the variant where a clause is satisfied when its literals do not all evaluate to the same value. Jansen and Pieterse showed [17, 18] that for $d \geq 3$, the d -NAE-SAT problem has a compression of size $\mathcal{O}(n^{d-1} \log n)$, but not of size $\mathcal{O}(n^{d-1-\varepsilon})$ unless $\text{NP} \subseteq \text{co-NP/poly}$. This example shows that the type of constraints affects the compressibility of a CSP.

The notion of a *constraint language* is used to rigorously analyze how the complexity of a CSP depends on the type of constraints. In this work, we will only consider CSPs over the Boolean domain: we work exclusively with Boolean constraints and constraint languages. A constraint is therefore a function of the form $f: \{0, 1\}^k \rightarrow \{0, 1\}$, where $k \geq 1$ is the *arity* of the constraint, also denoted as $\text{AR}(f)$. A constraint language Γ is a *finite* set of constraints. The input of the corresponding decision problem, denoted $\text{CSP}(\Gamma)$, consists of a set of constraint applications of the form $f(x_{j_1}, \dots, x_{j_{\text{AR}(f)}}) = 1$ over n common variables, where f is some constraint from Γ . The question is whether there is an assignment $\{x_1, \dots, x_n\} \rightarrow \{0, 1\}$ satisfying all the constraint applications.

In this terminology, Chen, Jansen, and Pieterse [9] characterized for all (Boolean) constraint languages Γ consisting of constraints of arity at most three, what the optimal compression size is for $\text{CSP}(\Gamma)$. Lagerkvist and Wahlström [25] gave universal-algebraic conditions on Γ which ensure that $\text{CSP}(\Gamma)$ has a compression of size $\mathcal{O}(n \log n)$, and a characterization is known of the constraint languages Γ_{sym} consisting entirely of *symmetric* functions for which $\text{CSP}(\Gamma_{\text{sym}})$ has a compression of near-linear size [9, §5]. Hence there is some understanding of the optimal compressibility of $\text{CSP}(\Gamma)$.

However, when we move from the question of whether *all* constraints can be satisfied to the task of maximizing the number of satisfied constraints (MAX CSP), the situation is much less understood. To the best of our knowledge, no non-trivial compressions are known for any MAX CSP(Γ), and no compression lower bounds are known for MAX CSP(Γ) other than those already implied from CSP(Γ). In this paper, we therefore analyze the compressibility of MAX CSP(Γ).

Before presenting our results, we briefly summarize the main algorithmic approach for compressing CSP(Γ) and illustrate why it fails completely for MAX CSP. Consider for example 3-NAE-SAT. The number of constraint applications in an n -variable instance of this problem can be reduced to $\mathcal{O}(n^2)$ without changing the solution space, which allows it to be encoded in $\mathcal{O}(n^2 \log n)$ bits. The *sparsification* to $\mathcal{O}(n^2)$ constraint applications is achieved by a linear-algebraic approach. Note that a not-all-equal constraint on variables $(x, y, z) \in \{0, 1\}^3$ is satisfied if and only if $x + y + z - xy - xz - yz - 1 = 0$. Observe that if $p_1(x_1, \dots, x_n) = 0, \dots, p_m(x_1, \dots, x_n) = 0$ are polynomial equalities which are satisfied by an assignment to x_1, \dots, x_n , then also $\sum_{i=1}^m \alpha_i \cdot p_i(x_1, \dots, x_n) = 0$ holds for any linear combination as determined by $\alpha_1, \dots, \alpha_m$. To sparsify a 3-NAE-SAT instance with this insight, proceed as follows. Transform each constraint c_i into an equality $p_i(x_1, \dots, x_n) = 0$ for a degree-2 polynomial p_i , substituting $1 - v$ for negated variables $\neg v$ in the constraint. This yields a system of equations of degree-2 polynomials in n variables, which have $\mathcal{O}(n^2)$ distinct monomials. The rank of a corresponding vector space is therefore $\mathcal{O}(n^2)$, which yields a basis of $\mathcal{O}(n^2)$ equalities such that all others can be expressed as their linear combinations. All constraints not corresponding to an element of this basis can be safely omitted from an instance of 3-NAE-SAT, since they will be automatically satisfied by any assignment that satisfies *all* basis constraints. This yields the claimed sparsification of $\mathcal{O}(n^2)$ constraints. Note, however, that this approach fails completely for the variant MAX 3-NAE-SAT: if an assignment *does not* satisfy all constraints of the basis, this does not give any satisfaction guarantees on the linearly-dependent constraints. Hence the sparsification approach for CSP(Γ) is not applicable for MAX CSP(Γ).

Our results. We provide a new route to compression for MAX CSP(Γ), and prove that the resulting compressions are *essentially optimal* for all constraint languages Γ , assuming $\text{NP} \not\subseteq \text{co-NP}/\text{poly}$. Our results characterize the optimal compressibility of all Boolean MAX CSPs in terms of degrees of characteristic polynomials, and uncover a wide range of MAX CSP(Γ) problems that admit a non-trivial compression. For a Boolean function $f: \{0, 1\}^k \rightarrow \{0, 1\}$, its *characteristic polynomial* is the *unique* k -variate multilinear polynomial $P_f(x)$ over \mathbb{R} that agrees with f on all $x \in \{0, 1\}^k$. The fact that this representation is unique is well-known (cf. [29]). For a constraint language Γ , define $\text{deg}(\Gamma) = \max_{f \in \Gamma} \text{deg}(P_f)$. We prove that $\text{deg}(\Gamma)$ characterizes the compressibility of MAX CSP(Γ).

To state our results precisely, we have to address a feature of the problem that is particular to the maximization variant: repetitions of constraint applications. While such repetitions are irrelevant in the CSP setting when all constraint applications have to be satisfied, they become relevant when maximizing the number of satisfied constraint applications. The standard approach in the MAX CSP literature is therefore to give each constraint application a positive integer weight value [11, 21]. The decision problem MAX CSP(Γ) then takes as input a system of Γ -constraint applications with weights from \mathbb{N} , and a threshold value t , and asks whether there is an assignment such that the weight of the satisfied constraint applications is at least t .

Let Γ be a (finite, Boolean) constraint language.¹ Our main positive result is the following.

► **Theorem 1.1.** *MAX CSP(Γ) parameterized by the number of variables n , with positive integer weights bounded by $n^{\mathcal{O}(1)}$, admits a compression of size $\mathcal{O}(n^{\deg(\Gamma)} \log n)$.*

In fact, we are even able to reduce any instance of MAX CSP(Γ) to an equivalent instance of the *same* problem, having $\mathcal{O}(n^{\deg(\Gamma)})$ weighted constraint applications. We prove matching lower bounds whenever MAX CSP(Γ) is NP-complete. It is known [10, 11, 21] that for inputs with positive integer weights, MAX CSP(Γ) is polynomial-time solvable if Γ is 0-valid, 1-valid, or 2-monotone (see Section 2.1), and NP-complete otherwise.

► **Theorem 1.2.** *If Γ is not 0-valid, 1-valid, or 2-monotone, then assuming $NP \not\subseteq co-NP/poly$, MAX CSP(Γ) parameterized by the number of variables n , with positive integer weights bounded by $n^{\mathcal{O}(1)}$, does not admit a compression of size $\mathcal{O}(n^{\deg(\Gamma)-\varepsilon})$ for any $\varepsilon > 0$.*

Our results uncover an interesting contrast in compressibility between decision CSPs and maximization CSPs. While both involve the analysis of the degrees of polynomials, the type of polynomials which is used differs, leading to differences in compressibility. For example, while d -NAE-SAT has a compression of size $\mathcal{O}(n^{d-1} \log n)$ for *all* $d \geq 3$, the corresponding MAX d -NAE-SAT problem with weights of absolute value $n^{\mathcal{O}(1)}$ has a compression of size $\mathcal{O}(n^{d-1} \log n)$ for *odd* $d \geq 3$, but no compression of size $\mathcal{O}(n^{d-\varepsilon})$ for *even* d . Another example is d -EXACT SAT, where we require exactly one literal in each clause to be true. Whereas d -EXACT SAT admits a compression of size $\mathcal{O}(n \log n)$ for every fixed d [9], we show that MAX d -EXACT SAT cannot be compressed to $\mathcal{O}(n^{d-\varepsilon})$ bits.

Techniques. On a high level, our results are obtained by combining two ingredients: (1) a characterization of the complexity of a constraint language as $\deg(\Gamma)$, via the degree of the characteristic polynomials, and (2) reductions between different problems MAX CSP(Γ) and MAX CSP(Γ') by implementing constraints of one language by combinations of constraints from the other. While both ingredients have been used in isolation [10, 11, 21, 26, 33], their combination is novel and is the key to understanding compressibility. To comprehend how characteristic polynomials help to compress an instance of MAX CSP(Γ), observe that since the characteristic polynomial gives 1 when a constraint is satisfied and 0 otherwise, the total value of satisfied constraint applications can be written as a weighted sum of applications of characteristic polynomials. If $\deg(\Gamma) = k$, then this weighted sum contains $\mathcal{O}(n^k)$ distinct monomials. An instance can therefore be compressed by expanding this weighted sum, and storing the coefficient of each monomial. If all weights in the input instance are bounded by $n^{\mathcal{O}(1)}$, each coefficient will have value $n^{\mathcal{O}(1)}$ and can therefore be encoded in $\mathcal{O}(\log n)$ bits.

Our lower bounds are obtained by parameterized reductions between MAX CSPs in which the number of variables does not grow significantly. By a careful analysis of the terms of the characteristic polynomial, we show that if $\deg(\Gamma) = \deg(\Gamma')$, then constraint applications from Γ can effectively be simulated by combinations of constraints from Γ' . Here, we use the framework of *implementations* from an earlier work [21]. Since the characteristic polynomial of d -CNF clauses has degree d , this yields a reduction from d -CNF-SAT to MAX CSP(Γ) for $\deg(\Gamma) = d$ that preserves the asymptotic size of the variable set, therefore transferring the cited lower bound for d -CNF-SAT [14] to MAX CSP(Γ). A similar reduction is also used for our positive results, to turn the monomial-based compression sketched above into a self-reduction which outputs an instance of the original problem.

¹ While some recent work on sparsification for CSPs allows infinite constraint languages Γ [18], they are not interesting from our perspective as $\deg(\Gamma) = +\infty$.

Consequences for exponential-time algorithms. The framework we develop for parameterized reductions among MAX CSPs also has consequences for exponential-time algorithms, which we believe to be of independent interest. The *MAX 3-SAT HYPOTHESIS* [26] states that MAX 3-CNF-SAT with n variables cannot be solved in time $\mathcal{O}(2^{(1-\varepsilon)n})$ for any $\varepsilon > 0$ (cf. [1, 5]). Our reductions imply that this hypothesis is *equivalent* to the version where MAX 3-CNF-SAT is replaced by MAX CSP(Γ) for any constraint language Γ with $\deg(\Gamma) = 3$ in which negated literals can be expressed (§2.2). In particular, the MAX 3-SAT hypothesis is equivalent to the statement that MAX E3-LIN cannot be solved in time $\mathcal{O}(2^{(1-\varepsilon)n})$. What is more, for any $k \geq 2$, our reductions uncover an equivalence class of NP-hard problems whose optimal exponential-time running times coincide with the one for MAX k -SAT.

Related work. Representations of Boolean functions by characteristic polynomials have been studied frequently in the literature [3, 4, 26, 28, 29, 31, 32] revealing, e.g., a relation between the degree of the representation and the decision tree complexity [29]. Algorithms for CSPs via their characteristic polynomials were first given by Williams [33]. He used the split-and-list technique to give accelerated exponential-time algorithms for MAX 2-SAT and MAX CSP(Γ) for $\deg(\Gamma) = 2$. In recent work, Lincoln, Williams, and Vassilevska Williams [26] give an exponential-time split-and-list reduction from MAX CSP(Γ) for $\deg(\Gamma) = k$ to detecting an ℓ -hyperclique in a k -uniform hypergraph, for $\ell > k$, in support of the (k, ℓ) -HYPERCLIQUE HYPOTHESIS, which states that detecting such a hyperclique in an n -vertex input requires time $n^{\ell-o(1)}$ on a Word-RAM with $\mathcal{O}(\log n)$ -bit words. If this hypothesis fails for some k and ℓ , their reduction implies that each MAX CSP(Γ) problem with $\deg(\Gamma) = k$ can be solved in time $\mathcal{O}(2^{(1-\varepsilon)n})$ for some $\varepsilon > 0$. Their reductions run in exponential time and are very different from ours.

Alon et al. [2] used a different representation of Boolean functions as polynomials in the work on MAX r -SAT parameterized above the guarantee. Here, the goal is to find an assignment satisfying at least $((2^r - 1)m + k)/2^r$ clauses, where m is the total number of clauses and k is the parameter. They have shown that the problem is FPT and admits a polynomial kernel.

Organization. We begin with Section 2 containing the necessary definitions and properties of CSPs, including the implementation framework. In Section 3 we explain the idea of representing constraints by polynomials and provide an algebraic background for our reductions. It is followed by Section 4, where the notion of a reduction between constraint systems is formalized, and the main reductions are presented. It serves as a toolbox for proving the main results for compression (Section 5) and exponential-time algorithms (Section 6). The proofs of statements indicated with (★) can be found in the full version [19].

2 Preliminaries

For a set S and integer $d \geq 0$, let $\binom{S}{d}$ be the set of all size- d subsets of S . We use $[n]$ as a shorthand for $\{1, \dots, n\}$. A k -ary constraint is a function $f: \{0, 1\}^k \rightarrow \{0, 1\}$. We refer to k as the arity of f , denoted $\text{AR}(f)$. We always assume that the domain is Boolean. A constraint f is satisfied by an input $s \in \{0, 1\}^k$ if $f(s) = 1$. A constraint language (sometimes called constraint family) Γ is a finite collection of constraints $\{f_1, f_2, \dots, f_\ell\}$, potentially with different arities. A *constraint application*, of a k -ary constraint f to a set of n Boolean variables, is a triple $\langle f, (i_1, i_2, \dots, i_k), w \rangle$, where the indices $i_j \in [n]$ select k of the n Boolean variables to whom the constraint is applied, and w is a weight, described formally below. The variables can be repeated in a single application.

► **Definition 2.1.** A constraint system is a pair $CS(\Gamma, \mathbb{W})$, where Γ is a constraint language and \mathbb{W} (for weight range) is either \mathbb{Z} or \mathbb{N} . An instance (or formula) of $CS(\Gamma, \mathbb{W})$ is a set of constraint applications from Γ over a common set of variables, each application having a weight from \mathbb{W} .

We denote the number of constraint applications in a formula Φ by $|\Phi|$ and the sum of absolute values of all weights in Φ by $\|\Phi\|$. For an assignment x , that is, a mapping from the set of variables to $\{0, 1\}$, the integer $\Phi(x)$ is the sum of weights of the constraint applications satisfied by x .

In the decision problem $\text{MAX CSP}(\Gamma, \mathbb{W}, c)$ we are given a formula Φ from $CS(\Gamma, \mathbb{W})$ over n variables such that $\|\Phi\| \leq n^c$, together with the integer t , and we ask if there is an assignment x such that $\Phi(x) \geq t$. We specify the constant c to be accurate about the specific decision problems for which we show hardness results (the formal definitions of parameterized problems and compression are given in Section 2.4). When it does not lead to confusion, e.g., when some property holds for all c , we refer to this family of problems shortly as $\text{MAX CSP}(\Gamma, \mathbb{W})$. Whenever we use the \mathcal{O} -notation, we do it with respect to a fixed problem, that is, we treat Γ and c as constants. The most commonly studied case is expressed by $\mathbb{W} = \mathbb{N}$ [13, 21, 27], where the weights can be interpreted as repetitions of constraint applications. It is important to make this distinction because it can be the case that $\text{MAX CSP}(\Gamma, \mathbb{N})$ is polynomially solvable whereas $\text{MAX CSP}(\Gamma, \mathbb{Z})$ is NP-hard [20]. Although our main reduction framework works for $\mathbb{W} = \mathbb{Z}$, we are able to transfer the compression lower bounds to the case $\mathbb{W} = \mathbb{N}$ as long as $\text{MAX CSP}(\Gamma, \mathbb{N})$ is NP-hard.

Another decision problem that is related to constraint systems is $\text{EXACT CSP}(\Gamma, \mathbb{W})$, where we ask whether there is an assignment for which the satisfied weights sum up exactly to a given integer [26, 33]. Even though we focus on the maximization variant, we formulate our reductions so that they could be employed for other problems over constraint systems or larger weight domains.

2.1 Types of constraints

We start by formally defining the most important constraints and constraint properties. They allow us to formulate the dichotomy theorem for MAX CSP . We use the Boolean notation for negation, i.e., $\neg x = 1 - x$ for $x \in \{0, 1\}$.

- A constraint is trivial if it is either always 1 or always 0 regardless of the arguments.
- The unary constraints T and F are given by $T(x) = x$ and $F(x) = \neg x$.
- OR_k and AND_k are k -ary constraints, such that $\text{OR}_k(x_1, \dots, x_k) = \bigvee_{i=1}^k x_i$ and analogously $\text{AND}_k(x_1, \dots, x_k) = \bigwedge_{i=1}^k x_i$. The Not-All-Equal constraint is defined as $\text{NAE}_k(x_1, \dots, x_k) = \text{OR}_k(x_1, \dots, x_k) \wedge \text{OR}_k(\neg x_1, \dots, \neg x_k)$.
- XOR_k is a k -ary constraint defined as $\text{XOR}_k(x_1, \dots, x_k) = x_1 + \dots + x_k \pmod{2}$. We abbreviate $\text{XOR} = \text{XOR}_2$.
- A constraint f is 0-valid (resp. 1-valid) if $f(0, 0, \dots, 0) = 1$ (resp. $f(1, 1, \dots, 1) = 1$).
- A constraint f is 2-monotone if $f(x_1, x_2, \dots, x_k) = (x_{i_1} \wedge x_{i_2} \wedge \dots \wedge x_{i_p}) \vee (\neg x_{j_1} \wedge \neg x_{j_2} \wedge \dots \wedge \neg x_{j_q})$, for some $p, q \geq 0$, $(p, q) \neq (0, 0)$, i.e., f is equivalent to a DNF-formula with at most two terms: one containing only positive literals and the other containing only negative literals.
- A constraint f is C-closed (complementation-closed) if for every assignment $x \in \{0, 1\}^{\text{AR}(f)}$, $f(x) = f(\bar{x})$, where \bar{x} stands for the bit-wise complement of x .
- A constraint f is symmetric if for any two assignments $x_1, x_2 \in \{0, 1\}^{\text{AR}(f)}$ having the same number of ones, it holds that $f(x_1) = f(x_2)$.

A constraint language Γ is called 0-valid, 1-valid, 2-monotone, C-closed, or symmetric, if all non-trivial constraints in Γ satisfy the respective property. We call Γ *non-trivial* if it contains at least one non-trivial constraint. This regime is convenient for formulating the fundamental dichotomy theorem for Boolean MAX CSP. For our purposes it is only important that APX-hardness entails NP-hardness.

► **Theorem 2.2** ([21, Theorem 2.11], cf. [10]). *MAX CSP(Γ, \mathbb{N}) is solvable in polynomial time if Γ is either 0-valid, 1-valid, or 2-monotone. Otherwise, the problem is APX-hard.*

2.2 Closures of constraint languages

In some CSPs we are allowed to write constraint applications containing constants or negations of variables, which makes them more convenient to process. We formalize these properties with the notion of a *language closure*.

► **Definition 2.3.** *Let f be a k -ary constraint and let g be a d -ary constraint. We say that g is expressible by f with constants if the identity $g(x_1, x_2, \dots, x_d) = f(\xi_1, \xi_2, \dots, \xi_k)$ holds for a tuple $(\xi_1, \xi_2, \dots, \xi_k)$, where each ξ_j is either a variable x_i for some $i \in [d]$ or one of the constants 0, 1.*

We say that g is expressible by f with literals if such an identity holds for a tuple $(\xi_1, \xi_2, \dots, \xi_k)$, where each ξ_j is a literal: either a variable x_i or its negation $\neg x_i$ for $i \in [d]$.

For a constraint language Γ we introduce the following closures:

- the language $\Gamma^{T,F}$ contains all functions expressible by $f \in \Gamma$ with constants,
- the language Γ^{LIT} contains all functions expressible by $f \in \Gamma$ with literals,
- the language Γ^{NEG} is the *negation-wise closure* of Γ , i.e., $\Gamma^{NEG} = \bigcup_{f \in \Gamma} \{f, \neg f\}$.

It is easy to see that the closures satisfy $(\Gamma^{T,F})^{T,F} = \Gamma^{T,F}$, $(\Gamma^{LIT})^{LIT} = \Gamma^{LIT}$, $(\Gamma^{NEG})^{NEG} = \Gamma^{NEG}$. We will be particularly interested in those constraint languages in which negated literals can be expressed, as in, e.g., d -CNF-SAT or d -NAE-SAT. These are the languages that satisfy $\Gamma = \Gamma^{LIT}$. Below we present examples on how to express important CSPs using our definitions.

- d -CNF-SAT = CSP($\Gamma_{d\text{-SAT}}$) for $\Gamma_{d\text{-SAT}} = \{\text{OR}_d\}^{LIT}$,
- d -NAE-SAT = CSP($\{\text{NAE}_d\}^{LIT}$),
- MAX Ed-LIN = MAX CSP($\{\text{XOR}_d\}^{NEG}, \mathbb{N}$),
- MAX CUT = MAX CSP($\{\text{XOR}\}, \mathbb{N}$),
- MAX DiCUT = MAX CSP($\{f\}, \mathbb{N}$) for $f(x_1, x_2) = x_1 \wedge \neg x_2$.

2.3 Constraint implementations

We describe the technique behind Theorem 2.2 [21]. The idea is to *implement* a constraint f by a collection of other constraints, so that satisfying f is equivalent to maximizing the number of satisfied constraints in that collection. It allows us to express formulas from MAX CSP(Γ_1, \mathbb{N}) by those from MAX CSP(Γ_2, \mathbb{N}), as long as constraints in Γ_1 can be implemented by those in Γ_2 .

The caveat is that each implementation may introduce new auxiliary variables whereas for our purposes we need reductions that increase the number of variables only by a multiplicative constant. Therefore the reductions by Khanna et al. [21] do not transfer compressibility bounds; we will use the implementations in a different way. On the other hand, our reductions do not preserve approximation factors.

► **Definition 2.4** ([21, Definition 3.1]). *A collection of unit-weighted constraint applications C_1, C_2, \dots, C_m over a set of variables $x = \{x_1, x_2, \dots, x_p\}$ called primary variables and $y = \{y_1, y_2, \dots, y_q\}$ called auxiliary variables, is an α -implementation of a constraint $f(x)$ for a positive integer α if the following conditions hold.*

1. Any assignment to x and y satisfies at most α constraint applications from C_1, C_2, \dots, C_m .
2. $\forall x$ such that $f(x) = 1$, $\exists y$ such that exactly α constraint applications are satisfied.
3. $\forall x, y$ such that $f(x) = 0$, at most $\alpha - 1$ constraint applications are satisfied.

An α -implementation is called *strict* if for every assignment of the primary variables x such that $f(x) = 0$, there exists an assignment of the auxiliary variables y such that exactly $\alpha - 1$ constraint applications are satisfied.

We say that a constraint language Γ implements a constraint f if there exists an α -implementation of f using constraints of Γ for some constant α . We use $\Gamma \implies f$ to denote that Γ implements f and $\Gamma \xrightarrow{s} f$ when Γ strictly implements f . The above notation is also extended to allow the target to be a family of constraints. The following lemma encapsulates a toolbox of implementations which we will rely on.

► **Lemma 2.5.** *If Γ is a non-trivial constraint language such that*

1. Γ is C-closed and not 0-valid (or equivalently not 1-valid), then $\Gamma \implies \text{XOR}$,
2. Γ is neither 0-valid, 1-valid, nor C-closed, then $\Gamma \implies \{T, F\}$,
3. Γ is neither 0-valid, 1-valid, nor 2-monotone, then $\Gamma \implies \text{XOR}$.

In order to prove it, we need to refer to several statements from [21], beginning from the transitivity of strict implementations. Then we restate three lemmas that imply points (1, 2) directly, and point (3) is obtained via transitivity.

► **Lemma 2.6** ([21, Lemma 3.5]). *If $\Gamma_1 \xrightarrow{s} \Gamma_2$ and $\Gamma_2 \xrightarrow{s} \Gamma_3$, then $\Gamma_1 \xrightarrow{s} \Gamma_3$.*

► **Lemma 2.7** ([21, Lemma 4.5]). *Let f be a non-trivial constraint which is C-closed and is not 0-valid (or equivalently not 1-valid). Then $\{f\} \xrightarrow{s} \text{XOR}$.*

► **Lemma 2.8** ([21, Lemma 4.6]). *Let f_0, f_1 , and g be non-trivial constraints, possibly identical, which are not 0-valid, not 1-valid, and not C-closed, respectively. Then $\{f_0, f_1, g\} \xrightarrow{s} \{T, F\}$.*

► **Lemma 2.9** ([21, Lemma 4.11]). *Let f be a constraint which is not 2-monotone. Then $\{f, T, F\} \xrightarrow{s} \text{XOR}$.*

Proof of Lemma 2.5. Claims (1, 2) follow from Lemmas 2.7 and 2.8, respectively. To see claim (3), first observe that if Γ is C-closed, then we can again use Lemma 2.7. Otherwise, by Lemma 2.8 we have $\Gamma \xrightarrow{s} \{T, F\}$. Next, we take advantage of transitivity (Lemma 2.6) and implement XOR with Lemma 2.9. ◀

2.4 Parameterized complexity

A *parameterized problem* is a decision problem in which every input has an associated positive integer *parameter* that captures its complexity in some well-defined way. In our study of CSPs we use the number of variables as the parameter, but other choices have been considered [16, 22, 23]. For a parameterized problem $A \subseteq \Sigma^* \times \mathbb{N}$, a decision problem $B \subseteq \Sigma^*$, and a function $f: \mathbb{N} \rightarrow \mathbb{N}$, a *compression* of A into B of size f is an algorithm that, on input $(x, k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance $y \in \Sigma^*$ such that $(x, k) \in A$ if and only if $y \in B$, and such that $|y| \leq f(k)$. A *kernelization* algorithm of size f for problem A reduces any instance (x, k) to an $f(k)$ -sized equivalent instance of the *same problem* in polynomial time.

3 Characteristic polynomials

In this section we provide the technique necessary for expressing one constraint system by another without introducing too many auxiliary variables. This insight is based on interpreting constraints as multilinear polynomials.

► **Definition 3.1.** For a k -ary constraint $f: \{0, 1\}^k \rightarrow \{0, 1\}$ its characteristic polynomial P_f is the unique k -ary multilinear polynomial over \mathbb{R} satisfying $f(x) = P_f(x)$ for any $x \in \{0, 1\}^k$.

It is easy to construct P_f . First define $P_s(x_1, x_2, \dots, x_k) = \prod_{i=1}^k R_i^s(x_i)$ for a vector $s \in \{0, 1\}^k$, where $R_i^s(x) = x$ if $s_i = 1$ and $R_i^s(x) = 1 - x$ otherwise. Formally, P_s is given by the sequence of coefficients obtained by expanding all parentheses; they are all integers. It holds that $P_s(s) = 1$, while $P_s(x) = 0$ for any $x \neq s$. For a constraint f its characteristic polynomial is given as $P_f(x_1, x_2, \dots, x_k) = \sum_{s: f(s)=1} P_s(x_1, x_2, \dots, x_k)$. It is known that no other multilinear polynomial can take identical values on $\{0, 1\}^k$ [29, 33]. This also means we can interchangeably analyze polynomials as formal objects and as functions on $\{0, 1\}^k$.

► **Observation 3.2.** The coefficients of any characteristic polynomial P_f are integers.

Let $\deg(f) = \deg(P_f)$ and $\deg(\Gamma) = \max_{f \in \Gamma} \deg(f)$. For a k -ary constraint f we refer to the coefficient at the unique k -ary monomial in P_f as the leading coefficient. The leading coefficient is non-zero if and only if $\deg(P_f) = k$. If g is expressible by f with literals, then we can obtain P_g from P_f by replacing each literal with negation $\neg x_i$ by $1 - x_i$ and expanding the parentheses within monomials. If g is expressible by f with constants, then we just substitute 0 or 1 for particular variables and remove monomials containing 0. These transformations imply $\deg(\Gamma^{T,F}) = \deg(\Gamma^{LIT}) = \deg(\Gamma^{NEG}) = \deg(\Gamma)$.

As an example, consider MAX 3-NAE-SAT. The function $\text{NAE}_3(x_1, x_2, x_3)$ has the degree-2 characteristic polynomial $x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3$, which allows us to construct a compression for MAX 3-NAE-SAT of size $\mathcal{O}(n^2 \log n)$ by summing coefficients at all $\mathcal{O}(n^2)$ monomials. On the other hand, $\text{OR}_2(x_1, x_2) = x_1 + x_2 - x_1x_2 = \text{NAE}_3(x_1, x_2, 0)$, which indicates that solving MAX 3-NAE-SAT should not be easier than MAX 2-CNF-SAT. We will formalize these arguments in the next section.

We show that the set of characteristic polynomials of all constraints expressible by f with constants spans the linear space of multilinear polynomials over \mathbb{Q} with degrees at most $\deg(f)$. We first prove that this set contains polynomials of all degrees up to $\deg(f)$ and then use them to express a basis of the linear space.

► **Lemma 3.3 (★).** Let f be a k -ary constraint. For any $1 \leq d \leq \deg(f)$ there exists a d -ary constraint g expressible by f with constants, such that its characteristic polynomial P_g has degree exactly d , i.e., its leading coefficient is non-zero.

► **Lemma 3.4.** Let f be a non-trivial constraint and P be a multilinear polynomial over \mathbb{Q} on ℓ variables of degree $0 \leq d \leq \deg(f)$. There exists a sequence of constraint applications $\langle f_i, (j_i^1, \dots, j_i^{\text{AR}(f_i)}), \alpha_i \rangle_{i=1}^M$ on ℓ variables, where each constraint f_i is expressible by f with constants, and $\alpha_i \in \mathbb{Q}$, such that the following polynomial identity holds.

$$P(x_1, \dots, x_\ell) = \sum_{i=1}^M \alpha_i \cdot P_{f_i}(x_{j_i^1}, \dots, x_{j_i^{\text{AR}(f_i)}}).$$

63:10 Optimal Polynomial-Time Compression for Boolean Max CSP

Before proving this claim, let us demonstrate it on a less obvious example than the one from above with NAE_3 and OR_2 . Let $P(x_1, x_2, x_3)$ be the characteristic polynomial of the constraint $\text{OR}_3(x_1, x_2, \neg x_3)$, derived using the method described below Definition 3.1:

$$\begin{aligned} P(x_1, x_2, x_3) &= x_1x_2x_3 + (1-x_1)x_2x_3 + x_1(1-x_2)x_3 + x_1x_2(1-x_3) \\ &\quad + (1-x_1)x_2(1-x_3) + x_1(1-x_2)(1-x_3) + (1-x_1)(1-x_2)(1-x_3) \\ &= 1 - x_3 + x_1x_3 + x_2x_3 - x_1x_2x_3. \end{aligned}$$

We will represent it with characteristic polynomials from $\{\text{EX}_3\}^{T,F}$, where $\text{EX}_3(x_1, x_2, x_3) = 1$ if and only if exactly one variable is 1. Its characteristic polynomial Q is given as:

$$\begin{aligned} Q(x_1, x_2, x_3) &= x_1(1-x_2)(1-x_3) + (1-x_1)x_2(1-x_3) + (1-x_1)(1-x_2)x_3 \\ &= x_1 + x_2 + x_3 - 2x_1x_2 - 2x_1x_3 - 2x_2x_3 + 3x_1x_2x_3. \end{aligned}$$

We can express P as the following linear combination where, e.g., $Q(x_1, x_2, 0)$ is the characteristic polynomial for $\text{EX}_3(x_1, x_2, 0)$, which is a binary constraint expressible by EX_3 with constants:

$$\begin{aligned} P(x_1, x_2, x_3) &= -\frac{1}{3}Q(x_1, x_2, x_3) + \frac{1}{3}Q(x_1, x_2, 0) - \frac{1}{6}Q(x_1, x_3, 0) - \frac{1}{6}Q(x_2, x_3, 0) \\ &\quad + \frac{1}{6}Q(x_1, 0, 0) + \frac{1}{6}Q(x_2, 0, 0) - \frac{1}{3}Q(x_3, 0, 0) + Q(1, 0, 0). \end{aligned}$$

Proof of Lemma 3.4. We proceed by induction over the degree of P . Since f is non-trivial, it admits a satisfying assignment s_T . If P is constant, then $P(x) = \alpha \cdot f(s_T)$ for some $\alpha \in \mathbb{Q}$. Suppose now $d = \deg(P) \geq 1$. For each $S \in \binom{[d]}{d}$ let α_S denote the (potentially zero) coefficient in P at the monomial $\prod_{i \in S} x_i$. By Lemma 3.3 there is a d -ary constraint f_d , which is expressible by f with constants and $\deg(P_{f_d}) = d$. Let β_d denote the leading coefficient of P_{f_d} . The polynomial

$$P'(x_1, \dots, x_\ell) = P(x_1, \dots, x_\ell) - \sum_{\substack{\{i_1, \dots, i_d\} = S \in \binom{[d]}{d} \\ i_1 < \dots < i_d}} \frac{\alpha_S}{\beta_d} \cdot P_{f_d}(x_{i_1}, \dots, x_{i_d})$$

has no monomials of degree d , since each term in the sum subtracts exactly one of them. The polynomial P' has degree at most $d-1$, so we can apply the induction hypothesis to it and represent P' as a linear combination of characteristic polynomials of constraints from $\{f\}^{T,F}$. We obtain the claim by adding these polynomials to the sum above. \blacktriangleleft

Since f_i and P_{f_i} coincide as functions on $\{0, 1\}^{\text{AR}(f_i)}$, Lemma 3.4 allows us to represent any constraint of degree at most $\deg(f)$ as a linear combination of constraints from $\{f\}^{T,F}$.

► Proposition 3.5. *Let g, f be constraints, such that f is non-trivial and $\deg(g) \leq \deg(f)$. There exists a sequence of constraint applications $\langle f_i, (j_i^1, \dots, j_i^{\text{AR}(f_i)}) \rangle_{i=1}^M$ on $\text{AR}(g)$ variables, where each constraint f_i is expressible by f with constants, and $\alpha_i \in \mathbb{Q}$, such that $g(x_1, \dots, x_{\text{AR}(g)}) = \sum_{i=1}^M \alpha_i \cdot f_i(x_{j_i^1}, \dots, x_{j_i^{\text{AR}(f_i)}})$ for all Boolean assignments.*

This resembles the idea of implementation, where we additionally equip constraints with (potentially negative) rational weights, but does not require introducing new variables.

4 Reductions between constraint systems

We first formalize our notion of reduction. The objects we work with are constraint systems and the reductions between them imply analogous relations between the associated decision problems. The reduction is crafted in such a way that it preserves the numbers of variables and constraints up to a constant factor, and the total weight up to a polynomial factor.

► **Definition 4.1.** *A linear transformation from a constraint system $CS(\Gamma_1, \mathbb{W}_1)$ to another constraint system $CS(\Gamma_2, \mathbb{W}_2)$ is a polynomial-time procedure that given a formula Φ_1 of $CS(\Gamma_1, \mathbb{W}_1)$ over n_1 variables and integer t_1 , returns a formula $\Phi_2 \in CS(\Gamma_2, \mathbb{W}_2)$ over n_2 variables and integer t_2 , so that the following conditions hold:*

1. $n_2 = \mathcal{O}(n_1)$,
2. $|\Phi_2| = \mathcal{O}(|\Phi_1| + n_1)$,
3. $\|\Phi_2\| \leq \|\Phi_1\| \cdot n_1^{\mathcal{O}(1)}$,
4. $\exists_x \Phi_1(x) = t_1 \iff \exists_y \Phi_2(y) = t_2$,
5. $\exists_x \Phi_1(x) \geq t_1 \iff \exists_y \Phi_2(y) \geq t_2$.

If there exists a linear transformation from $CS(\Gamma_1, \mathbb{W}_1)$ to $CS(\Gamma_2, \mathbb{W}_2)$, we write concisely $CS(\Gamma_1, \mathbb{W}_1) \leq_{LIN} CS(\Gamma_2, \mathbb{W}_2)$. If (1) can be replaced with the stronger condition $n_2 = n_1 + \mathcal{O}(1)$, we call the transformation additive and write $CS(\Gamma_1, \mathbb{W}_1) \leq_{ADD} CS(\Gamma_2, \mathbb{W}_2)$.

Before moving forward, let us explain the importance of linear and additive transformations. We formulate two claims, which follow from the properties in Definition 4.1.

► **Proposition 4.2.** *If $CS(\Gamma_1, \mathbb{W}_1) \leq_{LIN} CS(\Gamma_2, \mathbb{W}_2)$ and $MAX\ CSP(\Gamma_2, \mathbb{W}_2, c)$ admits a compression of size $\mathcal{O}(n^d)$, then $MAX\ CSP(\Gamma_1, \mathbb{W}_1, c - \mathcal{O}(1))$ also admits a compression of size $\mathcal{O}(n^d)$.*

► **Proposition 4.3.** *If $CS(\Gamma_1, \mathbb{W}_1) \leq_{ADD} CS(\Gamma_2, \mathbb{W}_2)$ and $MAX\ CSP(\Gamma_2, \mathbb{W}_2, c)$ admits an algorithm with running time $T(n)$, then $MAX\ CSP(\Gamma_1, \mathbb{W}_1, c - \mathcal{O}(1))$ admits an algorithm with running time $T(n + \mathcal{O}(1))$.*

In particular, additive transformations preserve running times of the form $2^{(1-\varepsilon)n} n^{\mathcal{O}(1)}$.

► **Lemma 4.4 (★).** *Linear transformations (resp. additive transformations) are transitive.*

We continue with two simple additive transformations, which will allow us to use negations of constraints as an alternative to setting negative weights.

► **Lemma 4.5 (★).** *For every constraint language Γ we have*

1. $CS(\Gamma^{NEG}, \mathbb{Z}) \leq_{ADD} CS(\Gamma, \mathbb{Z})$,
2. $CS(\Gamma^{NEG}, \mathbb{Z}) \leq_{ADD} CS(\Gamma^{NEG}, \mathbb{N})$.

The rest of this section contains four lemmas that form a chain of reductions from $CS(\Gamma_1, \mathbb{Z})$ to $CS(\Gamma_2, \mathbb{N})$, which is valid as long as $\deg(\Gamma_1) \leq \deg(\Gamma_2)$ and $MAX\ CSP(\Gamma_2, \mathbb{N})$ is NP-hard. First, we translate Proposition 3.5 into the language of additive transformations. In the proof we justify that one can replace rational coefficients with integer ones.

► **Lemma 4.6 (★).** *Let Γ_1, Γ_2 be non-trivial constraint languages satisfying $\deg(\Gamma_1) \leq \deg(\Gamma_2)$. Then $CS(\Gamma_1, \mathbb{Z}) \leq_{ADD} CS(\Gamma_2^{T,F}, \mathbb{Z})$.*

Now, we formalize an intuitive notation for combining formulas. Consider two formulas Φ_1, Φ_2 over the sets of variables V_1, V_2 , respectively, which might have a non-empty intersection. We define the sum of these formulas, $\Phi_1 + \Phi_2$, over the set of variables $V_1 \cup V_2$ by

taking the union of their sets of constraint applications and merging pairs of applications that share the same constraint and the same tuple of variables, i.e., replacing the pair with a single application with a weight being the sum of the respective weights. For an integer α , the formula $\alpha \cdot \Phi$ has the same constraint applications as Φ , with weights multiplied by α .

► **Lemma 4.7.** *Let Γ be a non-trivial constraint language, which is neither 0-valid nor 1-valid. Then $CS(\Gamma^{T,F}, \mathbb{Z}) \leq_{ADD} CS(\Gamma, \mathbb{Z})$.*

Proof. Given a formula $\Phi \in CS(\Gamma^{T,F}, \mathbb{Z})$, we add two auxiliary variables x_T, x_F and we translate each constraint application $\langle \hat{f}, (j_1, \dots, j_k), w \rangle$, where \hat{f} is expressible by $f \in \Gamma$ with constants, into an application of f by replacing constants 0, 1 with variables x_T, x_F . Let us refer to this formula as $\Phi_1 \in CS(\Gamma, \mathbb{Z})$ and note that $|\Phi_1| = |\Phi|$ and $\|\Phi_1\| = \|\Phi\|$.

In the next step, we will use implementations to impose particular conditions on x_T, x_F . We refer to x_T, x_F and all the new variables introduced within the implementations as auxiliary. Let $a = \mathcal{O}(1)$ denote their number. In the new formula we assume that the first n variables x_1, \dots, x_n are the primary variables and x_{n+1}, \dots, x_{n+A} are auxiliary. For $x \in \{0, 1\}^{n+A}$ let $x|_n$ stand for the projection on the first n coordinates.

Assume first that Γ is not C-closed. Then by Lemma 2.5, point (2), we know that Γ α_1 -implements function T and α_2 -implements function F for some integers α_1, α_2 . Let $\alpha = \alpha_1 + \alpha_2$. We implement constraint applications $T(x_T)$ and $F(x_F)$, that is, we construct formulas $\Phi_T, \Phi_F \in CS(\Gamma, \mathbb{N})$ over the set of auxiliary variables, such that satisfying α constraint applications in $\Phi_T + \Phi_F$ is only possible when $x_T = 1$ and $x_F = 0$. Let $W = 2 \cdot \|\Phi\| + 1$. We define $\Phi_2 = \Phi_1 + W \cdot \Phi_T + W \cdot \Phi_F$, that is, we copy all the constraint applications from Φ_1 and add the applications from $\Phi_T + \Phi_F$ with weights multiplied by W . Recall that we have $(-\|\Phi\|) \leq \Phi(x) \leq \|\Phi\|$ for all x . By the definition of implementation, any assignment to Φ_2 which does not satisfy $x_T = 1$ or $x_F = 0$ has value at most $(\alpha - 1) \cdot W + \|\Phi\| < \alpha W - \|\Phi\|$. If the assignment x satisfies $x_T = 1, x_F = 0$, it holds that $\Phi_2(x) = \alpha W + \Phi(x|_n) \geq \alpha W - \|\Phi\|$.

Now, if Γ is C-closed, then by Lemma 2.5, point (1), Γ α -implements function XOR for some constant α . We implement $\text{XOR}(x_T, x_F)$, that is, we construct formula $\Phi_{\text{XOR}} \in CS(\Gamma, \mathbb{N})$ over the set of auxiliary variables, such that satisfying α constraint applications in Φ_{XOR} is only possible when $\text{XOR}(x_T, x_F) = 1$. As before, we define $\Phi_2 = \Phi_1 + W \cdot \Phi_{\text{XOR}}$, where $W = 2 \cdot \|\Phi\| + 1$. Any assignment to Φ_2 which does not satisfy $\text{XOR}(x_T, x_F)$ has value at most $(\alpha - 1) \cdot W + \|\Phi\| < \alpha W - \|\Phi\|$. For an assignment x satisfying $x_T = 1, x_F = 0$, it holds that $\Phi_2(x) = \alpha W + \Phi(x|_n) \geq \alpha W - \|\Phi\|$. It might also be the case that $x_T = 0, x_F = 1$. Then, by C-closedness we have $\Phi_2(x) = \Phi_2(\bar{x}) = \alpha W + \Phi(\bar{x}|_n) \geq \alpha W - \|\Phi\|$, where \bar{x} is the bit-wise complement of x .

We summarize the transformation properties for both considered cases: the new number of variables is $n + \mathcal{O}(1)$, $|\Phi_2| = |\Phi| + \mathcal{O}(1)$, and $\|\Phi_2\| = \|\Phi\| \cdot \mathcal{O}(1)$. If $t < -\|\Phi\|$, then we know that all assignments x satisfy $\Phi(x) > t$ and in such case we could return an empty formula over a singleton variable set (so the only possible value is 0) and set threshold $t' = -1$: this is an equivalent instance. To see properties (4, 5) observe that, assuming $t \geq -\|\Phi\|$, $\Phi(x) = t$ (resp. $\Phi(x) \geq t$) holds for some assignment x iff. $\Phi_2(y) = t'$ (resp. $\Phi_2(y) \geq t'$) holds for some assignment y , where $t' = \alpha W + t$. ◀

► **Corollary 4.8.** *Let Γ_1, Γ_2 be non-trivial constraint languages such that $\deg(\Gamma_1) \leq \deg(\Gamma_2)$ and Γ_2 is neither 0-valid nor 1-valid. Then $CS(\Gamma_1, \mathbb{Z}) \leq_{ADD} CS(\Gamma_2, \mathbb{Z})$.*

So far we have established a relation between Γ_1 and Γ_2 , which allows us to transform one constraint system to another by adding only a constant number of new variables. However, it works only when we allow negative weights. The next two lemmas explain how to get rid of them, so that the hardness results can be applied to the natural setting with only non-negative weights.

► **Lemma 4.9 (★)**. *If Γ is a non-trivial constraint language, then $CS(\Gamma, \mathbb{Z}) \leq_{ADD} CS(\Gamma^{LIT}, \mathbb{N})$.*

► **Lemma 4.10 (★)**. *Suppose non-trivial Γ is neither 0-valid, 1-valid, nor 2-monotone. Then $CS(\Gamma^{LIT}, \mathbb{N}) \leq_{LIN} CS(\Gamma, \mathbb{N})$.*

In the proof of Lemma 4.9 we increase weights of all the constraint applications at once, in such a way that it changes each value of $\Phi(x)$ by the same quantity. The proof of Lemma 4.10 is similar in spirit to that of Lemma 4.7, but this time the implementation framework is used to implement a negated copy of each variable. As all the transformation above are linear, by transitivity we can summarize them into the following statement connecting constraint languages of the same degree.

► **Corollary 4.11**. *Let Γ_1, Γ_2 be non-trivial constraint languages such that $\deg(\Gamma_1) \leq \deg(\Gamma_2)$ and Γ_2 is neither 0-valid, 1-valid, nor 2-monotone. Then $CS(\Gamma_1, \mathbb{Z}) \leq_{LIN} CS(\Gamma_2, \mathbb{N})$.*

5 Consequences for compression

Having an upper bound on $\deg(\Gamma)$ already provides compression for $\text{MAX CSP}(\Gamma, \mathbb{Z}, c)$, since we can represent all constraint applications as polynomials, sum the coefficients at all $\mathcal{O}(n^{\deg(\Gamma)})$ monomials, and store the weights in $\mathcal{O}(\log n)$ bits. Corollary 4.11 transforms the monomial-representation back into a small equivalent instance of $\text{MAX CSP}(\Gamma)$.

► **Theorem 5.1** (Formalization of Theorem 1.1). *MAX CSP(Γ, \mathbb{N}, c) parameterized by the number of variables n admits a compression of size $\mathcal{O}(n^d \log n)$ for all c , where $d = \deg(\Gamma)$. Furthermore, there is a polynomial-time algorithm that reduces any instance of $\text{MAX CSP}(\Gamma, \mathbb{N}, c)$ to an equivalent instance of $\text{MAX CSP}(\Gamma, \mathbb{N}, c + \mathcal{O}(1))$ of size $\mathcal{O}(n^d \log n)$. The analogous statements for $\text{MAX CSP}(\Gamma, \mathbb{Z}, c)$ also hold.*

Proof. If Γ is either trivial, 0-valid, 1-valid, or 2-monotone, then $\text{MAX CSP}(\Gamma, \mathbb{N}, c)$ can be solved in polynomial time, so the compression is trivial. Suppose that it does not have any of these properties. We will prove both claims by compressing a formula $\Phi_1 \in CS(\Gamma, \mathbb{Z})$ on n variables into a formula $\Phi_2 \in CS(\Gamma, \mathbb{N})$ satisfying $|\Phi_2| = \mathcal{O}(n^d)$.

First we interpret each constraint application in Φ_1 as a polynomial of degree at most d . After summing these terms, we obtain a polynomial P with $\mathcal{O}(n^d)$ monomials, each associated with an integer weight of absolute value bounded by $\|\Phi_1\|$. A monomial $\prod_{i=1}^k x_i$ is the characteristic polynomial for the constraint $\text{AND}_k(x_1, \dots, x_k)$, therefore P can be treated as a formula of $CS(\Gamma_{d\text{-AND}}, \mathbb{Z})$ for $\Gamma_{d\text{-AND}}$ being the language consisting of functions AND_k for all $k \leq d$. Since $\deg(\Gamma_{d\text{-AND}}) = d$, we can apply Corollary 4.11 to obtain an equivalent formula $\Phi_2 \in CS(\Gamma, \mathbb{N})$ on $\mathcal{O}(n)$ variables, such that $|\Phi_2| = |\Phi_1| \cdot \mathcal{O}(1) + \mathcal{O}(n) = \mathcal{O}(n^d)$ and $\|\Phi_2\| = \|\Phi_1\| \cdot n^{\mathcal{O}(1)}$, so each weight can be stored in $\mathcal{O}(\log n)$ bits. ◀

The self-reduction in Theorem 5.1 is almost, but not quite, a kernelization: the formal decision problem we reduce to is not the same as the original one, due to the increase in weight values. Our lower bounds are based on reducing $\text{MAX } d\text{-CNF-SAT}$ to $\text{MAX CSP}(\Gamma, \mathbb{N})$ for $\deg(\Gamma) = d$. For $d \geq 3$ it is known that even the non-maximization variant $d\text{-CNF-SAT}$ does not admit a compression of size $\mathcal{O}(n^{d-\varepsilon})$ [14]. However, the 2-CNF-SAT problem is solvable in polynomial time and only its maximization version becomes NP-hard. We first note that $\text{MAX } 2\text{-CNF-SAT}$ also cannot have any non-trivial compression.

► **Lemma 5.2 (★)**. *MAX CSP($\Gamma_{2\text{-SAT}}, \mathbb{N}, 3$) does not admit a compression of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq \text{co-NP/poly}$.*

► **Theorem 5.3** (Formalization of Theorem 1.2). *Let non-trivial Γ be neither 0-valid, 1-valid, nor 2-monotone, and let $d = \deg(\Gamma)$. Then there is a constant c such that MAX CSP(Γ, \mathbb{N}, c) does not admit a compression of size $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{co-NP/poly}$.*

Proof. First observe that a characteristic polynomial of a constraint with $d = 1$ is linear. Since this polynomial is 0/1-valued, it can depend only on one variable, which means that the constraint is 2-monotone. Hence we can assume that $d \geq 2$.

We know that there is a constant c_d so that MAX CSP($\Gamma_{d\text{-SAT}}, \mathbb{N}, c_d$) does not admit compression of size $\mathcal{O}(n^{d-\varepsilon})$ unless $\text{NP} \subseteq \text{co-NP/poly}$: for $d = 2$ it is due to Lemma 5.2 and for $d \geq 3$ it follows from the lower bound for the non-maximization variant of d -CNF-SAT [14]. As noted in Proposition 4.2, if we had such a compression for MAX CSP(Γ, \mathbb{N}, c) with sufficiently large c , then by Corollary 4.11 it would transfer to MAX CSP($\Gamma_{d\text{-SAT}}, \mathbb{N}, c_d$). ◀

Applications to specific CSPs. Having established both the lower and the upper bound, we can refer to $\deg(\Gamma)$ as the *optimal compression exponent* for MAX CSP(Γ, \mathbb{N}). We are now equipped with a handy but powerful tool for determining the optimal compressibility of MAX CSP(Γ, \mathbb{N}) as this task reduces to computing the degrees of characteristic polynomials in Γ .

As an example, we apply this technique to compute the optimal compression exponent to the following three problems, which are all NP-hard for $k \geq 2$:

- MAX k -NAE-SAT = MAX CSP($\{\text{NAE}_k\}^{LIT}, \mathbb{N}$),
- MAX Ek -LIN = MAX CSP($\{\text{XOR}\}^{NEG}, \mathbb{N}$),
- MAX k -EXACT-SAT = MAX CSP($\{\text{EX}_k\}^{LIT}, \mathbb{N}$), where $\text{EX}_k(x_1, \dots, x_k) = 1$ if and only if there is exactly one 1 in (x_1, \dots, x_k) .

Since $\deg(\Gamma^{LIT}) = \deg(\Gamma^{NEG}) = \deg(\Gamma)$ it suffices to analyze the characteristic polynomials for functions NAE_k , XOR_k , and EX_k . Let $e_i(x_1, \dots, x_k)$ denote i -th elementary symmetric polynomial, i.e., the sum of all degree- i monomials on k variables.

$$\begin{aligned} \text{NAE}_k(x_1, \dots, x_k) &= \sum_{i=1}^{k-1} (-1)^{i-1} e_i(x_1, \dots, x_k) && \text{for odd } k, \\ \text{NAE}_k(x_1, \dots, x_k) &= \sum_{i=1}^k (-1)^{i-1} e_i(x_1, \dots, x_k) && \text{for even } k, \\ \text{XOR}_k(x_1, \dots, x_k) &= \sum_{i=1}^k (-2)^{i-1} e_i(x_1, \dots, x_k) && \text{for all } k, \\ \text{EX}_k(x_1, \dots, x_k) &= \sum_{i=1}^k i \cdot (-1)^{i-1} e_i(x_1, \dots, x_k) && \text{for all } k. \end{aligned}$$

It is easy to check these formulas using the binomial theorem. We present the argument for XOR_k as an example. Suppose the number of 1s in the vector (x_1, \dots, x_k) is ℓ . Then $e_i(x_1, \dots, x_k)$ equals $\binom{\ell}{i}$ for $i \leq \ell$ and 0 for $i > \ell$. The formula for $\text{XOR}_k(x_1, \dots, x_k)$ becomes $\sum_{i=1}^{\ell} (-2)^{i-1} \binom{\ell}{i} = (-\frac{1}{2}) \cdot (\sum_{i=0}^{\ell} (-2)^i \binom{\ell}{i} - 1) = (-\frac{1}{2}) \cdot ((-1)^{\ell} - 1)$ which is 1 for odd ℓ and 0 for even ℓ , as expected. By these identities we deduce that the optimal compression exponent for MAX k -NAE-SAT is k in the even case, $k - 1$ in the odd case, and the optimal compression exponent for both MAX Ek -LIN and MAX k -EXACT-SAT is k .

An example of a non-symmetric constraint with a non-trivial upper bound on its degree is f_k , with $\text{AR}(f_k) = 3^k$, defined recursively: $f_0(x) = x$, and

$$f_k(x_1, \dots, x_{3^k}) = \text{NAE}_3(f_{k-1}(x_1, \dots, x_{3^{k-1}}), f_{k-1}(x_{3^{k-1}+1}, \dots, x_{2 \cdot 3^{k-1}}), f_{k-1}(x_{2 \cdot 3^{k-1}+1}, \dots, x_{3^k})).$$

Because $\deg(\text{NAE}_3) = 2$, we have $\deg(f_k) = 2^k = \text{AR}(f_k)^{\log_3(2)}$ [29].

It is tempting to seek a concise characterization of Γ for which one can obtain a non-trivial bound on $\deg(\Gamma)$ and therefore a non-trivial compression for MAX CSP(Γ, \mathbb{N}), where by non-trivial we mean a bound of the form $\deg(f) \leq \text{AR}(f) - 1$ for functions depending on all the coordinates. By generalizing the argument for NAE_k , it can be shown that $\deg(f) \leq$

$\text{AR}(f) - 1$ when the number of vectors that satisfy f and have an even number of 1s, is equal to the number of satisfying vectors with an odd number of 1s. Unfortunately, as far as we are aware no characterization is known describing all Γ with a non-trivial bound on $\text{deg}(\Gamma)$, not even for symmetric polynomials induced by symmetric constraints. There exist some interesting partial results though, e.g., that if the number of variables k is a prime minus one then the degree is always k , and in general $\text{deg}(f) \geq k - \mathcal{O}(k^{0.548})$ [32]. On the other hand, there are infinitely many symmetric functions for which $\text{deg}(f) = \text{AR}(f) - 3$ [32]. When it comes to non-symmetric functions, there exist infinitely many examples with $\text{deg}(f) \leq \log(\text{AR}(f))$ [30], which is asymptotically the lowest upper bound possible [29].

Compression lower bound for negative weights. For the sake of completeness, we show that $\text{MAX CSP}(\Gamma, \mathbb{Z})$ admits an analogous classification as $\text{MAX CSP}(\Gamma, \mathbb{N})$ that is, whenever $\text{MAX CSP}(\Gamma, \mathbb{Z})$ is NP-hard, then the upper bound from Theorem 5.1 is essentially tight. The dichotomy theorem for $\mathbb{W} = \mathbb{Z}$ can be stated in a simpler manner, as the problem becomes NP-hard whenever $\text{deg}(\Gamma) \geq 2$ [20] and the case $\text{deg}(\Gamma) = 1$ reduces to linear function maximization. This dichotomy will follow also from the reduction below. First, we show that we can drop the assumption on the language being non 0-valid/1-valid in Corollary 4.8.

► **Lemma 5.4.** *Let Γ_1, Γ_2 be non-trivial constraint languages such that $\text{deg}(\Gamma_1) \leq \text{deg}(\Gamma_2)$. Then $\text{CS}(\Gamma_1, \mathbb{Z}) \leq_{\text{ADD}} \text{CS}(\Gamma_2, \mathbb{Z})$.*

Proof. Since f and $\neg f$ cannot be 0-valid (or 1-valid) at the same time, the language Γ_2^{NEG} is neither 0-valid nor 1-valid. As we also have $\text{deg}(\Gamma_2) = \text{deg}(\Gamma_2^{\text{NEG}})$, we can apply Corollary 4.8 to Γ_1 and Γ_2^{NEG} , obtaining $\text{CS}(\Gamma_1, \mathbb{Z}) \leq_{\text{ADD}} \text{CS}(\Gamma_2^{\text{NEG}}, \mathbb{Z})$. By Lemma 4.5, point (1), we get $\text{CS}(\Gamma_2^{\text{NEG}}, \mathbb{Z}) \leq_{\text{ADD}} \text{CS}(\Gamma_2, \mathbb{Z})$, which proves the claim. ◀

► **Theorem 5.5.** *Let non-trivial Γ be such that $d = \text{deg}(\Gamma) \geq 2$. Then there is a constant c such that $\text{MAX CSP}(\Gamma, \mathbb{Z}, c)$ does not admit a compression of size $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{co-NP/poly}$.*

Proof. From Lemma 5.4 we know that $\text{CS}(\Gamma_{d\text{-SAT}}, \mathbb{Z}) \leq_{\text{ADD}} \text{CS}(\Gamma, \mathbb{Z})$. Therefore an $\mathcal{O}(n^{d-\varepsilon})$ -size compression for $\text{MAX CSP}(\Gamma, \mathbb{Z}, c)$ with sufficiently large c entails the same for $\text{MAX CSP}(\Gamma_{d\text{-SAT}}, \mathbb{Z}, c - \mathcal{O}(1))$, which implies $\text{NP} \subseteq \text{co-NP/poly}$ (Lemma 5.2 for $d = 2$ and [14] for $d \geq 3$). ◀

6 Consequences for exponential-time algorithms

As mentioned before, our framework of reductions can be used to preserve the exponential running time as well. Namely, if $\text{CS}(\Gamma_1, \mathbb{W}_1) \leq_{\text{ADD}} \text{CS}(\Gamma_2, \mathbb{W}_2)$, then an algorithm for $\text{MAX CSP}(\Gamma_2, \mathbb{W}_2, c)$ with running time $T(n)$ entails an algorithm for $\text{MAX CSP}(\Gamma_2, \mathbb{W}_2, c - \mathcal{O}(1))$ with running time $T(n + \mathcal{O}(1))$. All the constructed transformations, except from $\text{CS}(\Gamma^{\text{LIT}}, \mathbb{N}) \leq_{\text{LIN}} \text{CS}(\Gamma, \mathbb{N})$ (Lemma 4.10), are additive and in particular they work as long as negative weights are allowed. Alternatively, we can take advantage of other properties of particular constraint languages to remove the negative weights.

► **Lemma 6.1.** *Let $d = \text{deg}(\Gamma) \geq 2$. Then $\text{CS}(\Gamma_{d\text{-SAT}}, \mathbb{N}) \leq_{\text{ADD}} \text{CS}(\Gamma, \mathbb{Z}) \leq_{\text{ADD}} \text{CS}(\Gamma_{d\text{-SAT}}, \mathbb{N})$, that is, these constraint systems are equivalent with respect to the relation \leq_{ADD} .*

63:16 Optimal Polynomial-Time Compression for Boolean Max CSP

Proof. We take advantage of the fact that $\Gamma_{d\text{-SAT}}$ can express negated literals, i.e., $(\Gamma_{d\text{-SAT}})^{LIT} = \Gamma_{d\text{-SAT}}$. We have the following cycle of reductions.

$$\begin{aligned} \text{CS}(\Gamma_{d\text{-SAT}}, \mathbb{Z}) &\leq_{ADD} \text{CS}(\Gamma_{d\text{-SAT}}, \mathbb{N}) && \text{Lemma 4.9 for } (\Gamma_{d\text{-SAT}})^{LIT} = \Gamma_{d\text{-SAT}} \\ &\leq_{ADD} \text{CS}(\Gamma, \mathbb{Z}) && \text{Lemma 5.4} \\ &\leq_{ADD} \text{CS}(\Gamma_{d\text{-SAT}}, \mathbb{Z}) && \text{Lemma 5.4} \quad \blacktriangleleft \end{aligned}$$

► **Theorem 6.2.** *For each $d \geq 2$ and any constant $\alpha > 1$ either all the following problems admit an $\alpha^n n^{\mathcal{O}(1)}$ algorithm for all c , or none of them do:*

1. MAX CSP($\Gamma_{d\text{-SAT}}, \mathbb{N}, c$),
2. MAX CSP(Γ, \mathbb{Z}, c) for any Γ with $\deg(\Gamma) = d$,
3. MAX CSP(Γ, \mathbb{N}, c) for any Γ with $\deg(\Gamma) = d$ such that $\Gamma^{NEG} = \Gamma$ or $\Gamma^{LIT} = \Gamma$.

Proof. As noted in Proposition 4.3, if $\text{CS}(\Gamma_1, \mathbb{W}_1) \leq_{ADD} \text{CS}(\Gamma_2, \mathbb{W}_2)$ and MAX CSP($\Gamma_2, \mathbb{W}_2, c$) admits an algorithm with running time $\alpha^n n^{\mathcal{O}(1)}$, then the same holds for MAX CSP($\Gamma_1, \mathbb{W}_1, c - \mathcal{O}(1)$). The equivalency between (1) and (2) has been proven in Lemma 6.1. Since (2) is more general than (3), it suffices to reduce (2) into (3). Lemma 4.5, point (2), provides the reduction $\text{CS}(\Gamma, \mathbb{Z}) \leq_{ADD} \text{CS}(\Gamma^{NEG}, \mathbb{N})$ for the case $\Gamma^{NEG} = \Gamma$ and Lemma 4.9 provides the reduction $\text{CS}(\Gamma, \mathbb{Z}) \leq_{ADD} \text{CS}(\Gamma^{LIT}, \mathbb{N})$ for the case $\Gamma^{LIT} = \Gamma$. ◀

First corollary of this theorem is that problems of form MAX CSP(Γ, \mathbb{Z}) are divided into equivalence classes with respect to the optimal running time. In particular, solving any MAX CSP(Γ, \mathbb{Z}) with $\deg(\Gamma) \geq 3$ in time $\mathcal{O}(2^{(1-\varepsilon)n})$ for any $\varepsilon > 0$ contradicts the MAX 3-SAT HYPOTHESIS. Also, the hypothesis remains equivalent if we replace MAX 3-CNF-SAT with MAX 3-LIN SAT or MAX 3-EXACT SAT with only positive weights, because their constraint languages satisfy $\Gamma^{NEG} = \Gamma$ and $\Gamma^{LIT} = \Gamma$, respectively. Another corollary is that improving the running time $\mathcal{O}(2^{\frac{\omega n}{3}})$ for MAX CUT or MAX DICUT with integer weights or MAX 3-NAE-SAT with positive weights would imply an analogous breakthrough for MAX 2-CNF-SAT.

Alman and Williams [1] have noted that it is not known how to improve the running time for MAX 3-CNF-SAT, even for instances with a linear number of clauses. They have therefore formulated a stronger hypothesis. The SPARSE MAX 3-SAT HYPOTHESIS states that there exists $c > 0$ such that MAX 3-CNF-SAT with cn clauses does not admit an $\mathcal{O}(2^{(1-\varepsilon)n})$ -time algorithm for any $\varepsilon > 0$. Similarly, their SPARSE MAX 2-SAT HYPOTHESIS states that one cannot beat running time $\mathcal{O}(2^{\frac{\omega n}{3}})$ for MAX 2-CNF-SAT with cn clauses. Observe that our reductions preserve the property of having $\mathcal{O}(n)$ different constraint applications (condition (2) in Definition 4.1). Therefore as long as we allow negative weights at constraint applications, we can replace MAX 2-CNF-SAT (resp. MAX 3-CNF-SAT) in this hypothesis with MAX CSP(Γ, \mathbb{Z}) for any constraint language Γ of degree 2 (resp. 3) to obtain an equivalent statement.

7 Conclusions and open problems

We have provided a complete characterization of the optimal compression for MAX CSP(Γ) in the case of a Boolean domain. A natural question arises about larger domains. Our approach does not transfer even to the case with a domain of size 3, since there is no unique way to represent functions $\{0, 1, 2\}^k \rightarrow \{0, 1\}$ as polynomials. One may consider, e.g., embedding to a Boolean domain or using non-multilinear polynomials, but it is not clear which approach leads to the optimal degree and how to find accompanying lower bounds.

On the exponential-time front, we have shown that MAX d -CNF-SAT is as hard as any MAX CSP of degree d as long as negative weights are allowed. Although we were able to get rid of the latter assumption in several cases, there is still a gap in this classification: does improving the running time for any degree- d MAX CSP(Γ, \mathbb{N}) imply an improvement for MAX d -CNF-SAT?

References

- 1 Josh Alman and Virginia Vassilevska Williams. OV Graphs Are (Probably) Hard Instances. In Thomas Vidick, editor, *Proceedings of the 11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 83:1–83:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2020.83.
- 2 Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011.
- 3 David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4(4):367–382, 1994. doi:10.1007/BF01263424.
- 4 Richard Beigel. The polynomial method in circuit complexity. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 82–95. IEEE Computer Society, 1993. doi:10.1109/SCT.1993.336538.
- 5 Karl Bringmann, Nick Fischer, and Marvin Künnemann. A Fine-Grained Analogue of Schaefer’s Theorem in P: Dichotomy of Exists^k-Forall-Quantified First-Order Graph Properties. In Amir Shpilka, editor, *Proceedings of the 34th Computational Complexity Conference, CCC 2019*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:27, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2019.31.
- 6 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In Chris Umans, editor, *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 319–330. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.
- 7 Andrei A. Bulatov, Martin Grohe, Phokion G. Kolaitis, and Andrei A. Krokhnin, editors. *The Constraint Satisfaction Problem: Complexity and Approximability, 25.10. - 30.10.2009*, volume 09441 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. URL: <http://drops.dagstuhl.de/portals/09441/>.
- 8 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. *J. ACM*, 64(3):19:1–19:39, 2017. doi:10.1145/2822891.
- 9 Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Best-case and worst-case sparsifiability of Boolean CSPs. *Algorithmica*, 2020. Online first. doi:10.1007/s00453-019-00660-y.
- 10 Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995. doi:10.1006/jcss.1995.1087.
- 11 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001. doi:10.1137/1.9780898718546.
- 12 Víctor Dalmau, Andrei A. Krokhnin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable finite-valued CSPs. *J. Comput. Syst. Sci.*, 97:14–27, 2018. doi:10.1016/j.jcss.2018.03.003.
- 13 Vladimir Deineko, Peter Jonsson, Mikael Klasson, and Andrei Krokhnin. The approximability of MAX CSP with fixed-value constraints. *J. ACM*, 55(4), September 2008. doi:10.1145/1391289.1391290.
- 14 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.

- 15 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.
- 16 Gregory Z. Gutin and Anders Yeo. Parameterized constraint satisfaction problems: a survey. In Andrei A. Krokhn and Stanislav Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 179–203. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/DFU.Vol17.15301.7.
- 17 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.
- 18 Bart M. P. Jansen and Astrid Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. *TOCT*, 11(4):28:1–28:26, 2019. doi:10.1145/3349618.
- 19 Bart M. P. Jansen and Michal Włodarczyk. Optimal polynomial-time compression for Boolean Max CSP. *CoRR*, abs/2002.03443, 2020. arXiv:2002.03443.
- 20 Peter Jonsson and Andrei A. Krokhn. Maximum H -colourable subdigraphs and constraint optimization with arbitrary weights. *J. Comput. Syst. Sci.*, 73(5):691–702, 2007. doi:10.1016/j.jcss.2007.02.001.
- 21 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.
- 22 Stefan Kratsch, Dániel Marx, and Magnus Wahlström. Parameterized Complexity and Kernelizability of Max Ones and Exact Ones Problems. *TOCT*, 8(1):1:1–1:28, 2016. doi:10.1145/2858787.
- 23 Stefan Kratsch and Magnus Wahlström. Preprocessing of Min Ones problems: A dichotomy. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Proceedings of the 37th International Colloquium on Automata, Languages and Programming, ICALP 2010*, volume 6198 of *Lecture Notes in Computer Science*, pages 653–665. Springer, 2010. doi:10.1007/978-3-642-14165-2_55.
- 24 Andrei A. Krokhn and Stanislav Zivny, editors. *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. URL: <http://www.dagstuhl.de/dagpub/978-3-95977-003-3>.
- 25 Victor Lagerkvist and Magnus Wahlström. Kernelization of constraint satisfaction problems: A study through universal algebra. In J. Christopher Beck, editor, *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming, CP 2017*, volume 10416 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2017. doi:10.1007/978-3-319-66158-2_11.
- 26 Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, SODA '18, pages 1236–1252, USA, 2018. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975031.80.
- 27 Konstantin Makarychev and Yury Makarychev. Approximation Algorithms for CSPs. In Andrei Krokhn and Stanislav Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 287–325. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol17.15301.287.
- 28 M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.
- 29 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994. doi:10.1007/BF01263419.
- 30 Hans Ulrich Simon. A tight $\Omega(\log \log n)$ -bound on the time for parallel RAM's to compute nondegenerated boolean functions. *Information and Control*, 55(1-3):102–106, 1982. doi:10.1016/S0019-9958(82)90477-6.
- 31 Gábor Tardos and David A. Mix Barrington. A lower bound on the mod 6 degree of the OR function. *Computational Complexity*, 7(2):99–108, 1998. doi:10.1007/PL00001597.

- 32 Joachim von Zur Gathen and James R. Roche. Polynomials with two values. *Combinatorica*, 17(3):345–362, September 1997. doi:10.1007/BF01215917.
- 33 R. Ryan Williams. *Algorithms and Resource Requirements for Fundamental Problems*. PhD thesis, Carnegie Mellon University, USA, 2007. AAI3274191.
- 34 Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983. doi:10.1016/0304-3975(83)90020-8.
- 35 Dmitry Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 331–342. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.38.