The Complexity of Promise SAT on Non-Boolean Domains

Alex Brandts
Department of Computer Science, University of Oxford, UK
alex.brandts@cs.ox.ac.uk

Marcin Wrochna
Department of Computer Science, University of Oxford, UK
marcin.wrochna@cs.ox.ac.uk

Stanislav Živný
Department of Computer Science, University of Oxford, UK
standa.zivny@cs.ox.ac.uk

Abstract

While 3-SAT is NP-hard, 2-SAT is solvable in polynomial time. Austrin, Guruswami, and Håstad [FOCS'14/SICOMP'17] proved a result known as "$(2+\varepsilon)$-SAT is NP-hard". They showed that the problem of distinguishing k-CNF formulas that are g-satisfiable (i.e. some assignment satisfies at least g literals in every clause) from those that are not even 1-satisfiable is NP-hard if $\frac{g}{k} < \frac{1}{2}$ and is in P otherwise. We study a generalisation of SAT on arbitrary finite domains, with clauses that are disjunctions of unary constraints, and establish analogous behaviour. Thus we give a dichotomy for a natural fragment of promise constraint satisfaction problems (PCSPs) on arbitrary finite domains.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Constraint and logic programming

Keywords and phrases promise constraint satisfaction, PCSP, polymorphisms, algebraic approach, label cover

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.17

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available [9] at `https://arxiv.org/abs/1911.09065`.

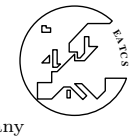
Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.
Alex Brandts: Royal Society Enhancement Award and NSERC PGS Doctoral Award
Stanislav Živný: Royal Society University Research Fellowship

Acknowledgements We would like to thank the anonymous reviewers for their comments.

1 Introduction

It is a classic result that while 3-SAT is NP-hard [12, 22], 2-SAT can be solved in polynomial-time [21]. Austrin, Guruswami, and Håstad [2] considered the promise problem $(1, g, k)$-SAT (for integers $1 \leq g \leq k$): given a k-CNF formula with the promise that there is an assignment that satisfies at least g literals in each clause, find an assignment that satisfies at least one literal in each clause. They showed that the problem is NP-hard if $\frac{g}{k} < \frac{1}{2}$ and in P otherwise. Viewing k-SAT as $(1, 1, k)$-SAT, this shows that, in a natural sense, the transition from tractability to hardness occurs just after 2 and not just before 3.

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 17; pp. 17:1–17:13

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


The *set-satisfiability* (SetSAT) problem generalises the Boolean satisfiability problem to larger domains and we prove that it exhibits an analogous hardness transition. As in (a, g, k)-SAT, for integer constants $1 \leq a \leq g \leq k$ and $1 \leq s < d$, an instance of the (a, g, k)-SetSAT problem is a conjunction of clauses, where each clause is a disjunction of k literals. However, variables $x_1, \ldots, x_n$ can take values in a larger domain $[d] = \{1, \ldots, d\}$, while literals take the form "$x_i \in S$", where S is any subset of the domain $[d]$ of size s. As in the Boolean case, an assignment $\sigma\colon \{x_1, \ldots, x_n\} \to [d]$ is g-satisfying if it satisfies at least g literals in every clause. In (a, g, k)-SetSAT with set size s and domain size d, given an instance promised to be g-satisfiable, we are to find an a-satisfying assignment. When $s = 1$ and $d = 2$ we recover Boolean promise SAT, whereas when $a = g = 1$ we recover the non-promise version of SetSAT.

The most natural case of SetSAT is when we allow *all* nontrivial unary constraints (sets) as literals, i.e., the case $s = d - 1$. (While we defined sets defining literals to have size exactly s, one can simulate sets of size at most s by replacing them with all possible supersets of size s; see the proof of [9, Proposition A.5]). More generally one could consider the problem restricted to any family of literals. Our work deals with the "folded" case: if a set S is available as a literal, then for all permutations of the domain π, $\pi(S)$ is also available as a literal. In this case only the cardinality of S matters, and in fact only the maximum available cardinality matters, so all such problems are equivalent to (a, g, k)-SetSAT, for some constants a, g, k, s, d.

1.1 Related work

Our main motivation to study SetSAT as a promise problem is the fact that it constitutes a natural fragment of so-called promise constraint satisfaction problems (PCSPs), which are problems defined by homomorphisms between relational structures (see Section 2 for more details). PCSPs were studied as early as in the classic work of Garey and Johnson [16] on approximate graph colouring, but a systematic study originated in the paper of Austrin et al. [2]. In a series of papers [5, 6, 7], Brakensiek and Guruswami linked PCSPs to the universal-algebraic methods developed for the study of non-uniform CSPs [4]. In particular, the notion of (weak) polymorphisms, formulated in [2], allowed some ideas developed for CSPs to be used in the context of PCSPs. The algebraic theory of PCSPs was then lifted to an abstract level by Barto, Bulín, Krokhin, and Opršal in [10, 3]. Consequently, this theory was used by Ficak, Kozik, Olšák, and Stankiewicz to obtain a dichotomy for symmetric Boolean PCSPs [15], thus improving on an earlier result from [6], which gave a dichotomy for symmetric Boolean PCSPs with folding. Further resent results on PCSPs include the work of Krokhin and Opršal [20], Brakensiek and Guruswami [8], and Austrin, Bhangale, and Potukuchi [1].

Variants of the Boolean satisfiability problem over larger domains have been defined using CNFs by Gil, Hermann, Salzer, and Zanuttini [17] and DNFs by Chen and Grohe [11] but, as far as we are aware, have not been studied as promise problems before.

1.2 Results

We completely resolve the complexity of (a, g, k)-SetSAT. As our main result, we show that the complexity of $(1, g, k)$-SetSAT depends only on the ratio $\frac{g}{k}$.

▶ **Theorem 1.** *$(1, g, k)$-SetSAT with set size s and domain size $s+1$ is solvable in polynomial time if $\frac{g}{k} \geq \frac{s}{s+1}$ and is NP-hard otherwise.*

Our result generalises the case of $(1, g, k)$-SAT, where $s = 1$ and the hardness threshold is $\frac{1}{2}$. The general case, when $a \neq 1$ or $d > s + 1$, follows by simple reductions (cf. [9, Corollary A.4]). The positive side of the theorem is proved by a simple randomised algorithm based on classical work of Papadimitriou [23], just as in the Boolean case. The main difficulty is in proving NP-hardness when the ratio $\frac{g}{k}$ is close to, but below $\frac{s}{s+1}$.

Following [2] and the more abstract algebraic framework of [3], the hardness proof relies on understanding polymorphisms, i.e., high-arity functions $f\colon [d]^n \to [d]$ which describe the symmetries of our computational problem. In the Boolean case, the proof of [2] relies on showing that every polymorphism depends on only a few variables (in other words, is a junta), and that this condition suffices for a reduction from the gap label cover problem. In our case, this condition does not hold, and neither do the various generalisations of it used in later work on PCSPs [15, 3, 20]. In fact, we show in [9, Section 6] that SetSAT has significantly richer, more robust polymorphisms, which makes the application of many such conditions impossible. Our main technical contribution is a new condition that guarantees an NP-hardness reduction from a multilayered variant of the gap label cover problem.

As in previous work, the combinatorial core of our NP-hardness results for SetSAT relies on identifying, in every polymorphism $f\colon [d]^n \to [d]$, a small set of distinguished coordinates. The rough idea is that a polymorphism encodes a 1-in-n choice analogously to the long code, and the reduction relies on being able to decode f with small ambiguity.

The set of distinguished coordinates could be, in the simplest case, those on which f depends (called *essential coordinates*) and, as shown in [2], a small set of essential coordinates is sufficient for hardness of $(1, g, k)$-SAT if $\frac{g}{k} < \frac{1}{2}$. More generally, the distinguished set S could be such that some partial assignment to S makes f constant (as a function of its remaining coordinates), or restricts the range of f (called *fixing* [2, 15] and *avoiding* [3] sets, respectively). As shown in [9, Section 6], the polymorphisms of SetSAT on non-Boolean domains do not have small sets of coordinates that are essential, fixing, or avoiding. Instead, in this paper we introduce the notion of a *smug set* of f. We say that a set $S \subseteq [n]$ is smug if for some input $(a_1, \dots, a_n)$ to f, the coordinates i whose values a_i agree with the output $f(a_1, \dots, a_n)$ are exactly those in S. We show that every polymorphism of SetSAT has a smug set of constant size (independent of n) and cannot have many disjoint smug sets.

In previous work, it was crucial that essential coordinates respect minors. We say that (an m-ary function) g is a *minor* of (an n-ary function) f if $g(x_1, \dots, x_m) \approx f(x_{\pi(1)}, \dots, x_{\pi(n)})$ for some $\pi\colon [n] \to [m]$ (that is, g is obtained from f by identifying or permuting coordinates of f, or introducing inessential coordinates). In that case, if S contains all essential coordinates of f, then $\pi(S)$ contains all essential coordinates of g. This does not hold for smug sets; instead, if S is a smug set of g, then its pre-image $\pi^{-1}(S)$ is a smug set of f. The pre-image may however be much larger. Still, these properties of smug sets are enough to guarantee that, in any sufficiently long chain of minors, if one chooses a random coordinate in a small smug set from each function in the chain, then for some two functions in the chain the choices will agree, respecting the minor relation between them with constant probability. We show that this condition is sufficient to obtain NP-hardness from a layered gap label cover problem. See Section 4 for details.

We note that several other properties of label cover variants were used before in the context of polymorphisms. Guruswami and Sandeep [18] use "smoothness" of NP-hard label cover instances (introduced by Khot [19]) so that a minor relation π needs to be respected only if it is injective on a small set S. This allows them to use sets S which are *weakly fixing*, i.e. the partial assignment to S which makes f constant does not necessarily have to assign the same value to all coordinates in S. Layered label cover was introduced

by Dinur, Guruswami, Khot, and Regev [13] to tighten the approximation hardness for hypergraph vertex cover. In the proof of hardness of hypergraph colouring by Dinur, Regev, and Smyth [14], as reinterpreted in [3], layered label cover is used to partition polymorphisms into an arbitrary constant number L of parts, so that only minors within one part need to be respected. This implies that in any chain of minors with $L+1$ functions, some two functions will be in the same part and hence the minor between them will be respected; our approach is hence similar, though apparently more general, in this aspect. Another feature used in [14, 3] is that the bound on the size of a set of special coordinates or on the number of disjoint such sets may be any subpolynomial function in n, not necessarily a constant. These different features of NP-hard label cover instances can be combined; however, this is not necessary for our result.

2 Preliminaries

Let $[n] = \{1, 2, \dots, n\}$. For a set A, we call $R \subseteq A^k$ a *relation* of arity $\mathrm{ar}(R) = k$ and $f\colon A^k \to B$ a function of arity $\mathrm{ar}(f) = k$.

We take the domain of the variables in SetSAT to be $[d]$ and for a fixed $s < d$ we identify each literal with the indicator function of some $S \subseteq [d], |S| = s$: $S(x) = \mathbb{1}[x \in S]$. For a SetSAT instance (or *formula*) with n variables $x_1, \dots, x_n$, an assignment to the variables is a function $\sigma : \{x_1, \dots, x_n\} \to [d]$. An assignment σ is called a g-satisfying assignment for an instance ϕ if σ satisfies at least g literals in every clause of ϕ. A 1-satisfying assignment is usually simply called a satisfying assignment. A formula is called g-satisfiable if there exists a g-satisfying assignment to its variables, and satisfiable if there exists a 1-satisfying assignment.

The SAT problem corresponds to the SetSAT problem with $d = 2$ and $s = 1$, so SetSAT does indeed generalise SAT. Note that every SetSAT instance is trivially unsatisfiable when $s = 0$ and satisfiable when $s = d$, so we exclude these cases in our analysis. We now give the formal definition of (a, g, k)-SetSAT.

▶ **Definition 2.** *Let $1 \le s < d$ and $1 \le a \le g \le k$. The (a, g, k)-SetSAT problem is the following promise problem. In the decision version, given a SetSAT instance where each clause has k literals, accept the instance if it is g-satisfiable and reject it if it is not a-satisfiable. In the search version, given a g-satisfiable SetSAT instance, find an a-satisfying assignment.*

We will prove hardness only for the decision version of (a, g, k)-SetSAT and tractability only for the search version. This suffices since the decision version of (a, g, k)-SetSAT is polynomial-time reducible to the corresponding search problem. This is discussed in [9, Appendix A], where it is also shown how to obtain simple hardness results for SetSAT. In particular, [9, Propositions A.1 and A.3] show that we can focus on the case of $(1, g, k)$-SetSAT with $d = s + 1$.

Promise CSPs

We describe how the SetSAT problem fits into the general framework of promise CSPs (PCSPs). For a more in-depth algebraic study of PCSPs, we refer the reader to [3].

A *relational structure* $\mathbf{A}$ is a tuple $(A; R_1, \dots, R_m)$ where each R_i is a relation on A. We say that two relational structures are *similar* if their relations have the same sequence of arities. A *homomorphism* between similar relational structures $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, , R_m^{\mathbf{A}})$ and $\mathbf{B} = (B; R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is a function $h : A \to B$ such that $(a_1, \dots, a_{\mathrm{ar}(R_i^{\mathbf{A}})}) \in R_i^{\mathbf{A}}$ implies $(h(a_1), \dots, h(a_{\mathrm{ar}(R_i^{\mathbf{A}})})) \in R_i^{\mathbf{B}}$ for all i. We denote this by $\mathbf{A} \to \mathbf{B}$.

▶ **Definition 3.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a pair of similar relational structures such that there is a homomorphism* $\mathbf{A} \to \mathbf{B}$*. The pair* $(\mathbf{A}, \mathbf{B})$ *is called the* template *of the* promise constraint satisfaction problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$*. The decision version of* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is as follows: given as input a relational structure* $\mathbf{C}$ *similar to* $\mathbf{A}$ *and* $\mathbf{B}$*, decide whether* $\mathbf{C}$ *admits a homomorphism to* $\mathbf{A}$*, or does not even admit a homomorphism to* $\mathbf{B}$*. The* promise *is that it is never the case that* $\mathbf{C} \to \mathbf{B}$ *but* $\mathbf{C} \not\to \mathbf{A}$*. The search problem asks to find a homomorphism* $\mathbf{C} \to \mathbf{B}$*, given that there exists a homomorphism* $\mathbf{C} \to \mathbf{A}$*.*

Since (a, g, k)-SetSAT is a PCSP where all relations have fixed arity k, it is possible to transform SetSAT instances from their CNF representation into the PCSP representation of Definition 3. Let f be a bijection between $[d^k]$ and the set of clauses containing k literals (ignoring the variables they contain). We can represent each SetSAT instance Ψ as a relational structure $\mathbf{C} = (C; R_1^{\mathbf{C}}, \dots, R_{d^k}^{\mathbf{C}})$, where $C = \{x_1, \dots, x_n\}$ is the set of variables appearing in Ψ and $R_i^{\mathbf{C}}$ is a k-ary relation corresponding to the clause $f(i)$. For each clause $(S_1(x_1) \vee \dots \vee S_k(x_k))$ of type $f(i)$ in Ψ, we add the tuple $(x_1, \dots, x_k)$ to $R_i^{\mathbf{C}}$, so that each $R_i^{\mathbf{C}}$ collects the tuples of variables appearing in clauses of type $f(i)$.

Now define $R_i^{\mathbf{A}}$ (respectively $R_i^{\mathbf{B}}$) to be the k-ary relation over $[d]$ containing $(a_1, \dots, a_k)$ if and only if $(a_1, \dots, a_k)$ g-satisfies (respectively a-satisfies) the clause $f(i)$ when the variable of the j-th literal of $f(i)$ is set to a_j, for $1 \le j \le k$. Let $\mathbf{A} = ([d], R_1^{\mathbf{A}}, \dots, R_{d^k}^{\mathbf{A}})$ and $\mathbf{B} = ([d], R_1^{\mathbf{B}}, \dots, R_{d^k}^{\mathbf{B}})$. Then (a, g, k)-SetSAT is precisely $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$: the identity function is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, a homomorphism $\mathbf{C} \to \mathbf{A}$ represents a g-satisfying assignment to Ψ, and a homomorphism $\mathbf{C} \to \mathbf{B}$ represents an a-satisfying assignment to Ψ.

Polymorphisms

The following concept from the algebraic study of PCSPs is central to our hardness result.

Let $f : A^m \to B$ be a function. We say that f is a *polymorphism* of the template $(\mathbf{A}, \mathbf{B})$ if, for $\bar{a}^1, \dots, \bar{a}^m \in R_i^{\mathbf{A}}$, we have that $f(\bar{a}^1, \dots, \bar{a}^m) \in R_i^{\mathbf{B}}$; here f is applied componentwise. We will denote by $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ the set of all polymorphisms of the template $(\mathbf{A}, \mathbf{B})$. A simple example of a polymorphism of every template with $A = B$ is a *projection*, which is a function $p_i^{(m)} : A^m \to B$ of the form $p_i^{(m)}(x_1, \dots, x_m) = x_i$. More generally, for every template, trivial polymorphisms are given by *dictators*, which are functions p of the form $p(x_1, \dots, x_m) = f(x_i)$, where f is a homomorphism from A to B.

In particular, $f : [d]^m \to [d]$ is a polymorphism of (a, g, k)-SetSAT if for every SetSAT clause C of width k and for every tuple $\bar{v}^1, \dots, \bar{v}^m \in [d]^k$ of g-satisfying assignments to C, we have that $f(\bar{v}^1, \dots, \bar{v}^m)$ is an a-satisfying assignment to C.

3 Tractability

How big must one make the fraction of satisfied literals in order for the SetSAT problem to become tractable? The following proposition shows that $\frac{s}{s+1}$ is sufficient.

▶ **Proposition 4.** *For* $1 \le s < d$ *and* $\frac{g}{k} \ge \frac{s}{s+1}$*,* $(1, g, k)$*-SetSAT is solvable in expected polynomial time.*

Proof. Algorithm 1 finds a satisfying assignment to a g-satisfiable formula in expected polynomial time. The algorithm and its analysis are based on [2, Proposition 6.1], which in turn is based on Papadimitriou's randomised algorithm for 2-SAT [23].

Algorithm 1 Randomised algorithm for $(1, g, k)$-SetSAT with $\frac{g}{k} \geq \frac{s}{s+1}$.

1: $x \leftarrow$ arbitrary assignment
2: **while** x does not satisfy input formula ϕ **do**
3: Arbitrarily pick a falsified clause C
4: Randomly choose from C a literal $S(x_i)$
5: Randomly choose a value for x_i so that $S(x_i)$ is satisfied
return x

Suppose that ϕ has a g-satisfying assignment x^*. Let x^t be the assignment obtained in iteration t of the algorithm, and let $D_t = \text{dist}(x^t, x^*)$, where $\text{dist}(x, y)$ is the Hamming distance between x and y. Since $D_t - D_{t-1} \in \{-1, 0, 1\}$ for every t, we have

$$\begin{aligned}\mathbb{E}(D_t - D_{t-1}) &= \mathbb{P}(D_t - D_{t-1} = 1) - \mathbb{P}(D_t - D_{t-1} = -1) \\ &\leq \frac{k-g}{k} - \frac{g}{k}\frac{1}{s} \leq 0 \quad \text{if and only if} \quad \frac{g}{k} \geq \frac{s}{s+1}.\end{aligned}$$

The sequence $D_0, D_1, \ldots$ is a random walk starting between 0 and n with steps either unbiased or biased toward 0. With constant probability, such a walk hits 0 within n^2 steps and so the probability that the algorithm fails to find a satisfying assignment within crn^2 steps is at most 2^{-r} for some constant c. ◀

▶ Remark 5. The proof of Proposition 4 can be modified to show that Algorithm 1 also finds a satisfying assignment when each literal corresponds to a set of size *at most* s. This makes sense intuitively, as smaller literals give the algorithm a better chance of setting x_i equal to x_i^* in Step 5.

We show that if $\frac{g}{k} \geq \frac{s}{s+1}$ then $(1, g, k)$-SetSAT has a specific family of polymorphisms that leads to a *deterministic* algorithm based on linear programming.

A function $f : A^m \to B$ is *symmetric* if $f(a_1, \ldots, a_m) = f(a_{\pi(1)}, \ldots, a_{\pi(m)})$ for all $a_1, \ldots, a_m \in A$ and all permutations π on $[m]$.

▶ **Definition 6.** *A symmetric function $f : [d]^m \to [d]$ is a* plurality *if*

$$f(x_1, \ldots, x_m) = \textit{argmax}_{a \in [d]}\{\#\ \textit{of occurrences of } a \textit{ in } (x_1, \ldots, x_m)\},$$

with ties broken in such a way that f is symmetric.

We will also use the fact that all polymorphisms of SetSAT are *conservative*; i.e., they always return one of their input values, as the following proposition shows.

▶ **Proposition 7.** *All polymorphisms of $(1, g, k)$-SetSAT are conservative.*

Proof. Let $f : [d]^m \to [d]$ be such that $f(a_1, \ldots, a_m) = b$ and $b \notin \{a_1, \ldots, a_m\}$. If S is a literal not containing b, then the clause $(S(x_1) \vee \ldots \vee S(x_k))$ is g-satisfied (even k-satisfied) by setting all x_i equal to any one of the a_j. Thus taking the m assignments $(x_1 = \cdots = x_k = a_j)_{1 \leq j \leq m}$ and applying f to each component, we get the assignment $x_1 = \cdots = x_k = b$ which clearly does not 1-satisfy the clause, and so f cannot be a polymorphism. ◀

▶ **Proposition 8.** *Let $s \geq 1$. If $\frac{g}{k} > \frac{s}{s+1}$ then every plurality function is a polymorphism of $(1, g, k)$-SetSAT. If $\frac{g}{k} = \frac{s}{s+1}$ then every plurality function of arity $m \not\equiv 0 \mod s+1$ is a polymorphism of $(1, g, k)$-SetSAT, and no symmetric function of arity $m \equiv 0 \mod s+1$ is a polymorphism of $(1, g, k)$-SetSAT.*

Proof. Let f be a plurality function of arity m. Given m g-satisfying assignments to a clause of width k, we are guaranteed to have at least mg satisfying values among the mk total values. Therefore there is a coordinate i, $1 \leq i \leq k$, containing at least $\lceil \frac{mg}{k} \rceil$ satisfying values, that is, at least $\lceil \frac{mg}{k} \rceil$ values not equal to the value b forbidden by the i-th literal of the clause. In order for f to be a polymorphism it suffices that $\lceil \frac{mg}{k} \rceil > \frac{s}{s+1}m$, since then b will appear fewer than $\frac{m}{s+1}$ times and will never be returned using the plurality rule. But $\frac{g}{k} > \frac{s}{s+1}$ is equivalent to $\frac{mg}{k} > \frac{s}{s+1}m$, and the latter implies that $\lceil \frac{mg}{k} \rceil > \frac{s}{s+1}m$, so f is a polymorphism.

In the case that $\frac{g}{k} = \frac{s}{s+1}$, the same argument works so long as $\frac{mg}{k}$ is not an integer, since by taking the ceiling we obtain a value strictly greater than $\frac{s}{s+1}m$. Since $\frac{g}{k} = \frac{s}{s+1}$, we have $\frac{g}{k}m = \frac{s}{s+1}m$ and this is an integer only if m is a multiple of $s+1$.

To show that there are no symmetric polymorphisms when $\frac{g}{k} = \frac{s}{s+1}$ and m is a multiple of $s+1$, note that this equality implies that k is divisible by $s+1$. Let M be the $(s+1) \times (s+1)$ matrix whose first row is $12 \cdots s+1$ and whose i-th row for $2 \leq i \leq s+1$ is obtained from the $(i-1)$-st row by shifting it cyclically to the left by one coordinate. We stack $\frac{k}{s+1}$ copies of M on top of each other and take $\frac{m}{s+1}$ copies of this stack side-by-side to form the $k \times m$ matrix M'. If f is symmetric, it returns the same value b when applied to each row of M'. Every column of M' satisfies exactly an $\frac{s}{s+1}$-fraction of the literals in a clause whose k literals all forbid b. On the other hand, the assignment produced by applying f to each row of M' does not even 1-satisfy this clause, so f is not a polymorphism. ◀

Proposition 8 has interesting consequences for solvability of $(1, g, k)$-SetSAT via linear programming relaxations. By [3, Theorem 7.9], $(1, g, k)$-SetSAT is solvable by the basic linear programming relaxation if $\frac{g}{k} > \frac{s}{s+1}$ (since there exist symmetric polymorphisms of all arities) but not solvable by the basic linear programming relaxation if $\frac{g}{k} = \frac{s}{s+1}$ (since there do not exist symmetric polymorphisms of all arities). By [8, Theorem 3.1], $(1, g, k)$-SetSAT is solvable by the combined basic linear programming and affine relaxation if $\frac{g}{k} \geq \frac{s}{s+1}$ (since there exist symmetric polymorphisms of infinitely many arities). We note that iterative rounding of the basic linear relaxation could also be used to get a deterministic algorithm as in [2].

4 Layered label cover and smug sets

An *ℓ-layered label cover* instance is a sequence of $\ell + 1$ sets $X_0, \ldots, X_\ell$ (called *layers*) of variables with range $[m]$, for some *domain size* $m \in \mathbb{N}$, and a set of constraints Φ. Each constraint is a function (often called a projection constraint) from a variable $x \in X_i$ to a variable in a further layer $y \in X_j$, $i < j$: that is, a function denoted $\phi_{x \to y}$ which is satisfied by an assignment $\sigma \colon X_0 \cup \cdots \cup X_\ell \to [m]$ if $\sigma(y) = \phi_{x \to y}(\sigma(x))$. A *chain* is a sequence of variables $x_i \in X_i$ for $i = 0, \ldots, \ell$ such that there are constraints $\phi_{x_i \to x_j}$ between them, for $i < j$. A chain is *weakly satisfied* if at least one of these constraints is satisfied.

The basis for our hardness result is the hardness of distinguishing fully satisfiable instances from those where no constant fraction of chains can be weakly satisfied. This follows by a simple adaptation of a reduction from the work of Dinur, Guruswami, Khot, and Regev [13], which we defer to the full version [9, Appendix B].

▶ **Theorem 9.** *For every $\ell \in \mathbb{N}$ and $\varepsilon > 0$, there is an $m \in \mathbb{N}$ such that it is NP-hard to distinguish ℓ-layered label cover instances with domain size m that are fully satisfiable from those where not even an ε-fraction of all chains is weakly satisfied.*

In order to use Theorem 9 to derive hardness for PCSPs, we use the algebraic approach: every PCSP is equivalent to a promise problem about satisfying minor conditions with polymorphisms. We give definitions first, following [3], to where we refer the reader for a more detailed exposition.

For $f\colon A^n \to B$, $g\colon A^m \to B$ and $\pi\colon [n] \to [m]$, we say that g is the *minor* of f obtained from π if

$$g(x_1, \dots, x_m) \approx f(x_{\pi(1)}, \dots, x_{\pi(n)}), \tag{1}$$

where $g \approx f$ means that the values of g and f agree on every input in A^m. We write $f \xrightarrow{\pi} g$ as a shorthand for (1). For $\pi\colon [n] \to [m]$, the expression $f \xrightarrow{\pi} g$ is called a *minor identity*.

A *minion* on a pair of sets (A, B) is a non-empty set of functions from A^n to B (for $n \in \mathbb{N}$) that is closed under taking minors.

A *bipartite minor condition* is a finite set Σ of minor identities where the sets of function symbols used on the left- and right-hand sides are disjoint. More precisely, Σ is a pair of disjoint sets U and V of function symbols of arity n and m, respectively, and a set of minor identities of the form $f \xrightarrow{\pi} g$, where $g \in U$, $f \in V$ and $\pi : [n] \to [m]$. A bipartite minor condition Σ is *satisfied* in a minion $\mathcal{M}$ if there is an assignment $\xi : U \cup V \to \mathcal{M}$ that assigns to each function symbol a function from $\mathcal{M}$ of the corresponding arity so that for every identity $f \xrightarrow{\pi} g$ in Σ, we have $\xi(f) \xrightarrow{\pi} \xi(g)$ in $\mathcal{M}$. A bipartite minor condition is called *trivial* if it is satisfied in every minion, or equivalently, in the minion consisting of all projections on $\{0, 1\}$. Since choosing a projection of arity n is the same as choosing an element of $[n]$, deciding whether a bipartite minor condition is trivial is the same as standard label cover.

We can now define the *promise satisfaction of a minor condition* problem. For a minion $\mathcal{M}$ and an integer m, $\mathrm{PMC}_{\mathcal{M}}(m)$ is the following promise problem: given a bipartite minor condition Σ that involves only symbols of arity at most m, the answer should be YES if Σ is trivial and NO if Σ is not satisfiable in $\mathcal{M}$ (the promise is that either of those two cases holds, i.e. an algorithm can behave arbitrarily otherwise). Barto et al. [3] show that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is log-space equivalent to $\mathrm{PMC}_{\mathcal{M}}(m)$, for $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ and m a constant depending on $\mathbf{A}$ only.

A final piece of notation before we prove a corollary of Theorem 9. A *chain of minors* is a sequence of the form $f_0 \xrightarrow{\pi_{0,1}} f_1 \xrightarrow{\pi_{1,2}} \dots \xrightarrow{\pi_{\ell-1,\ell}} f_\ell$. We shall then write $\pi_{i,j}\colon [\mathrm{ar}(f_i)] \to [\mathrm{ar}(f_j)]$ for the composition of $\pi_{i,i+1}, \dots, \pi_{j-1,j}$, for any $0 \le i < j \le \ell$. Note that $f_i \xrightarrow{\pi_{i,j}} f_j$.

▶ **Corollary 10** (of Theorem 9). *Let $\mathcal{M}$ be a minion. Suppose there are constants $k, \ell \in \mathbb{N}$ and an assignment of a set of at most k coordinates $\mathrm{sel}(f) \subseteq [\mathrm{ar}(f)]$ to every $f \in \mathcal{M}$ such that for every chain of minors $f_0 \xrightarrow{\pi_{0,1}} f_1 \xrightarrow{\pi_{1,2}} \dots \xrightarrow{\pi_{\ell-1,\ell}} f_\ell$, there are $0 \le i < j \le \ell$ such that $\pi_{i,j}(\mathrm{sel}(f_i)) \cap \mathrm{sel}(f_j) \neq \emptyset$. Then $\mathrm{PMC}_{\mathcal{M}}(m)$ is NP-hard, for m large enough. In particular, if $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.*

Proof. For ℓ, k as in the assumption, let $\varepsilon := \frac{1}{k}$ and let m be as given by Theorem 9. We reduce an ℓ-layered label cover instance by replacing each variable x with a symbol f_x of arity m and each constraint $\phi_{x\to y}\colon [m] \to [m]$ by the minor condition $f_x \xrightarrow{\phi_{x\to y}} f_y$. If the original instance was fully satisfiable, the new instance is trivial (i.e., fully satisfiable by projections).

If the constructed instance is satisfied by functions in the minion $\mathcal{M}$, we define an assignment to the original instance by selecting, for each variable x, a random coordinate from $\mathrm{sel}(f_x) \subseteq [m]$ (uniformly, independently). The assumption guarantees a set of constraints $\phi_{x\to y}$ such that (1) each chain contains at least one and (2) for each such constraint $\phi_{x\to y}$, we have $\phi_{x\to y}(\mathrm{sel}(f_x)) \cap \mathrm{sel}(f_y) \neq \emptyset$. The random choice then satisfies each of these constraints, and hence weakly satisfies each chain, with probability at least $\frac{1}{k} = \varepsilon$. The expected fraction of weakly satisfied chains is thus at least ε and a standard maximisation-of-expectation procedure deterministically finds an assignment which certifies this. ◀

The following definition is crucial to our results.

▶ **Definition 11.** *For a function $f\colon A^{\mathrm{ar}(f)} \to B$ we say that a set of coordinates $S \subseteq [\mathrm{ar}(f)]$ is a* smug *set if there is an input vector $\bar{v} \in A^{\mathrm{ar}(f)}$ such that $S = \{i \mid v_i = f(\bar{v})\}$.*

We will use the following to prove hardness of $(1, g, k)$-SetSAT.

▶ **Corollary 12.** *Let $\mathcal{M}$ be a minion. Suppose there are constants $k, \ell \in \mathbb{N}$ such that the following holds, for every $f \in \mathcal{M}$:*

- *f has a smug set of at most k coordinates,*
- *f has no family of more than ℓ (pairwise) disjoint smug sets,*
- *if $f \xrightarrow{\pi} g$ and S is a smug set of g, then $\pi^{-1}(S)$ is a smug set of f.*

Then $\mathrm{PMC}_{\mathcal{M}}(m)$ is NP-hard, for m large enough. In particular, if $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

Proof. For each $f \in \mathcal{M}$, we define $\mathrm{sel}(f)$ as a smug set of at most k coordinates, arbitrarily chosen (some such set exists by the first condition). Consider a chain $f_0 \xrightarrow{\pi_{0,1}} f_1 \xrightarrow{\pi_{1,2}} \dots \xrightarrow{\pi_{\ell-1,\ell}} f_\ell$. Suppose to the contrary that for each $0 \le i < j \le \ell$, $\pi_{i,j}(\mathrm{sel}(f_i))$ is disjoint from $\mathrm{sel}(f_j)$, or equivalently, that $\mathrm{sel}(f_i)$ is disjoint from $\pi_{i,j}^{-1}(\mathrm{sel}(f_j))$. This implies that $\pi_{0,i}^{-1}(\mathrm{sel}(f_i))$ is disjoint from $\pi_{0,i}^{-1}(\pi_{i,j}^{-1}(\mathrm{sel}(f_j))) = \pi_{0,j}^{-1}(\mathrm{sel}(f_j))$. That is, the sets $\pi_{0,i}^{-1}(\mathrm{sel}(f_i))$ for $i = 0 \dots \ell$ are pairwise disjoint. By the third condition they are smug sets of f_0. But by the second condition this is impossible. ◀

We note that in the proof of Corollary 12, the exact definition of "smug" is irrelevant, as long as it satisfies the above three conditions.

It is easy to check that the definition of "smug" satisfies the third condition for any functions $f \xrightarrow{\pi} g$, not necessarily polymorphisms. Indeed, if an input $\bar{v} \in A^{\mathrm{ar}(g)}$ to g gives a smug set $S = \{j \mid v_j = g(\bar{v})\}$, then the corresponding input $\bar{u} \in A^{\mathrm{ar}(f)}$ to f defined as $u_i := v_{\pi(i)}$ satisfies $f(\bar{u}) = g(\bar{v})$ and hence gives a smug set $\{i \mid u_i = f(\bar{u})\} = \{i \mid v_{\pi(i)} = g(\bar{v})\} = \{i \mid \pi(i) \in S\} = \pi^{-1}(S)$.

The definition of "smug" is particularly well-suited to our problem, because whether f is a polymorphisms or not depends only on its family of smug sets.

▶ **Lemma 13.** *Let $1 \le s$ and $1 \le g < k$. A function $f\colon [s+1]^m \to [s+1]$ is a polymorphism of $(1, g, k)$-SetSAT if and only if there is no multiset $S_1, \dots, S_k$ of smug sets of f, such that each coordinate $\ell \in [m]$ is contained in at most $k - g$ of them.*

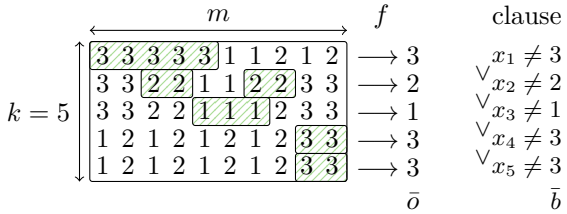


Figure 1 Illustration of Lemma 13. Smug sets $S \subseteq [m]$ are highlighted in each row.

Proof. A function $f\colon [s+1]^m \to [s+1]$ is not a polymorphism if and only if there is a clause of the form $x_1 \neq b_1 \vee \dots \vee x_k \neq b_k$ (for some column vector $\bar{b} \in [s+1]^k$) and a sequence of m column vectors $\bar{v}^1, \dots, \bar{v}^m \in [s+1]^k$ each of which g-satisfies the clause, but for which the vector $\bar{o} = f(\bar{v}^1, \dots, \bar{v}^m)$ (with f applied coordinatewise) does not even 1-satisfy the

clause. The latter is equivalent to saying that $o_i = b_i$ for $i \in [k]$, that is, applying f to the i-th row gives $f(v_i^1, \dots, v_i^m) = b_i$. The former is equivalent to saying that for each column $\bar{v}$ in $\bar{v}^1, \dots, \bar{v}^m$, the condition $v_i \neq b_i$ holds for at least g indices $i \in [k]$ of that column. The two are hence equivalent to saying that for each column $\bar{v}^\ell$, $\ell \in [m]$, the condition $v_i^\ell = f(v_i^1, \dots, v_i^m)$ holds for at most $k - g$ indices $i \in [k]$ in that column. In other words, the k row vectors $(v_i^1, \dots, v_i^m)$ for $i \in [k]$ have smug sets such that ℓ is contained in at most $k - g$ of these sets, for each coordinate $\ell \in [m]$. ◀

Checking the second condition for polymorphisms of our SetSAT problem is easy.

▶ **Lemma 14.** *For every polymorphism f of $(1, g, k)$-SetSAT with domain size $s + 1$, if $S_1, \dots, S_n$ are disjoint smug sets of f, then $n < \frac{k}{k-g}$.*

Proof. Suppose to the contrary that $n \geq \frac{k}{k-g}$. Then we can build a multiset containing each S_i up to $k - g$ times until we have exactly k in total. We thus obtain a multiset of k smug sets such that every coordinate is contained in at most $k - g$ of them. ◀

5 Finding small smug sets

It is easy to show NP-hardness when $\frac{g}{k} \leq \frac{1}{2}$ (cf. [9, Proposition A.8]). We now show a general reduction by finding a small smug set for $(1, g, k)$-SetSAT whenever $\frac{g}{k} < \frac{s}{s+1}$.

Consider a polymorphism $f \colon [s + 1]^m \to [s + 1]$ of $(1, g, k)$-SetSAT (with set size s and domain size $s + 1$).

▶ **Lemma 15.** *There exists a smug set of size at most $s-1$, or a family of s disjoint minimal smug sets $S_1, \dots, S_s$.*

Proof. Suppose that every smug set has size at least s. We show by induction on t that there is a family of t disjoint minimal smug sets $S_1, \dots, S_t$. Suppose we found $S_1, \dots, S_t$ for some $0 \leq t < s$ and we want to find S_{t+1}. Let T be a set containing one arbitrary coordinate from each S_i, $i = 1 \dots t$. Let $\bar{v} \in [s + 1]^m$ be the input vector with values $t + 2$ on T, i on $S_i \setminus T$ (for $i = 1 \dots t$) and $t + 1$ on the remaining coordinates $R := [m] \setminus (S_1 \cup \dots \cup S_t)$. Since $|T| \leq t < s$, T is not smug, so $f(\bar{v}) \neq t + 2$. By minimality, $S_i \setminus T$ are not smug for $i = 1 \dots t$, so $f(\bar{v}) \neq i$. Therefore, by conservativity of f (Proposition 7), the only remaining option is $f(\bar{v}) = t + 1$. Thus R is smug and disjoint from S_i. Taking S_{t+1} to be a minimal smug set contained in R proves the induction step. ◀

Together with Lemma 14, Lemma 15 already establishes (via Corollary 12) NP-hardness when $s \geq \frac{k}{k-g} = \frac{g}{k-g} + 1$ (equivalently, $\frac{g}{k} \leq \frac{s-1}{s}$): since there cannot be s disjoint smug sets, every polymorphism has a smug set of size at most $s - 1$. The proof in the general case, when $\frac{g}{k} < \frac{s}{s+1}$, extends this approach by first finding (assuming there are no small smug sets) disjoint minimal smug sets $S_1, \dots, S_s$, then exploiting the fact that each has a special coordinate whose removal makes it not smug, and using these coordinates to find further variants of each S_i with new special coordinates.

▶ **Lemma 16.** *Let $\frac{g}{k} < \frac{s}{s+1}$ (equivalently, $s > \frac{g}{k-g}$). Every polymorphism of $(1, g, k)$-SetSAT on s has a smug set of size at most g.*

Proof. Consider a polymorphism $f \colon [s + 1]^m \to [s + 1]$ of $(1, g, k)$-SetSAT. We prove by induction on t that there is a smug set of size at most $t - 1$, or there is a sequence of smug sets $S_1, \dots, S_t$ and a set T such that (see Figure 2):

(i) $|T| = t$ and $|T \cap S_i| = 1$ for $i = 1 \dots t$ (hence $S_i \cap T \neq S_{i'} \cap T$ for $i \neq i'$);
(ii) $S_i \setminus T$ is not smug for $i = 1 \dots t$;
(iii) $S_i \cap S_{i'} = \emptyset$ if $i \not\equiv i' \mod s$;
(iv) $S_i \supseteq S_{i-s} \setminus T$ for $i > s$.

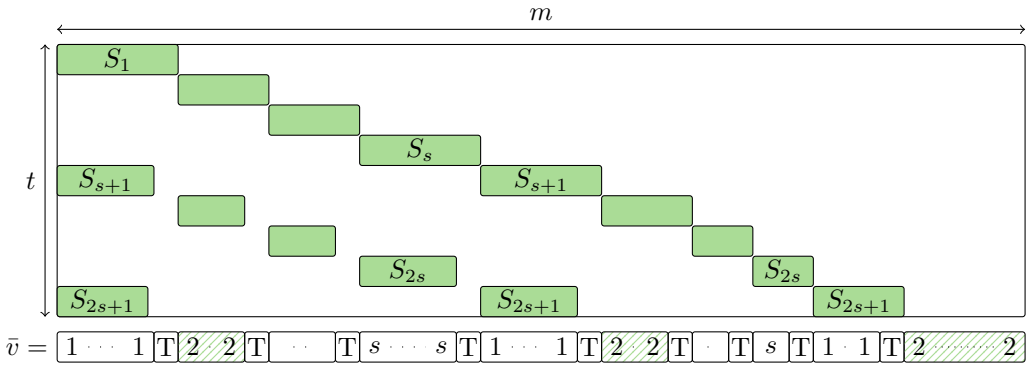


Figure 2 Illustration of smug sets obtained in the proof of Lemma 16. Each row represents one of the sets in the sequence $S_1, \dots, S_t$. The set T is formed by coordinates with a T and get values $s+1$. The vector $\bar{v}$ is used to find the next row S_{t+1}.

By Lemma 15 we can start with $t = s$ (by taking any T containing one coordinate from each S_i). Suppose the above is true for $t \geq s$ and let us prove the same for $t+1$. If there is a smug set of size at most t then we are done, so assume that T is not smug. Let $\bar{v} \in [s+1]^m$ be the input vector with value $s+1$ on T and different values from $\{1, \dots, s\}$ on $S_{t-i} \setminus T$ for $i = 0 \dots s-2$ and on the set of remaining coordinates $R := [m] \setminus (S_t \cup \dots \cup S_{t-s+2} \cup T)$. Then by (ii), R is smug.

Observe that R contains $S_{t-s+1} \setminus T$, because $S_t, \dots, S_{t-s+2}, T$ are disjoint from that set by (iii). We define S_{t+1} to be a minimal subset of R among smug sets containing $S_{t-s+1} \setminus T$. By (ii) $S_{t-s+1} \setminus T$ itself is not smug, so there exists some coordinate ℓ in $S_{t+1} \setminus S_{t-s+1}$. We choose it arbitrarily and set $T' := T \cup \{\ell\}$.

We claim that the sequence of smug sets $S_1, \dots, S_{t+1}$ and the set T' satisfy the above conditions. By minimality $S_{t+1} \setminus T'$ is not smug, so it satisfies (ii) and by definition it satisfies (iv). The set S_{t+1} is disjoint from $S_t, \dots, S_{t-s+2}, T$, because R was. It is also disjoint from S_i for $i \not\equiv t+1 \mod s$, because for every such i, $S_i \setminus T$ is contained in one of $S_t, \dots, S_{t-s+2}$; this proves (iii). In particular ℓ is not contained in any of these sets, and since it is not contained in S_{t-s+1}, it is in fact not contained in any S_i with $i < t+1$. Hence $|T'| = t$ and $|T' \cap S_i| = |T \cap S_i| = 1$ for $i < t+1$. Clearly also $|T' \cap S_{t+1}| = |\{\ell\}| = 1$. Therefore, (i) is satisfied, concluding the inductive proof.

Let us now consider sets as guaranteed above for $t = g+1$ (assuming there is no smug set of size at most g). Let $\bar{v} \in [s+1]^m$ be the input vector with value $i+1$ on $S_{t-i} \setminus T$ for $i = 0 \dots s-1$, and value $s+1$ on the remaining coordinates $R := ([m] \setminus (S_t \cup \dots \cup S_{t-s+1})) \cup T$. By (ii) the sets $S_{t-i} \setminus T$ are not smug, so R is smug. We claim that the multiset obtained from $\{S_1, \dots, S_t\}$ by adding $(k-g-1)$ copies of the set R contradicts Lemma 13: that is, each coordinate in $[m]$ is covered at most $k-g$ times by this multiset.

Consider first the coordinates contained in R. By definition of R, they are disjoint from $S_{t-i} \setminus T$ for $i = 0 \dots s-1$. By (iv), they are also disjoint from all sets $S_i \setminus T$ for $i = 0 \dots t$, because every such set is contained in one of the former. Hence if a coordinate in R is also contained in one of $S_1, \dots, S_t$, then it is contained in T and therefore in at most one of $S_1, \dots, S_t$, by (i). In total, it is thus covered at most $(k-g-1)+1 = k-g$ times.

Consider now coordinates outside of R. By (iii), they can be covered only by sets S_i with congruent indices $i \mod s$. Since $s > \frac{g}{k-g}$, we have $s(k-g) > g$, so there are $t = g + 1 \leq s(k-g)$ distinct indices in total in $\{1, \ldots, t\}$. Hence at most $k - g$ of them can be pairwise congruent to each other $\mod s$. Thus coordinates outside of R are also covered at most $k - g$ times. ◀

This concludes the proof that smug sets satisfy the first condition of Corollary 12 for polymorphisms of $(1, g, k)$-SetSAT with set size s and domain size $s + 1$, assuming $\frac{g}{k} < \frac{s}{s+1}$. Therefore, the problem is NP-hard. The full dichotomy then follows by simple reductions, see [9, Corollary A.4].

References

1 Per Austrin, Amey Bhangale, and Aditya Potukuchi. Improved inapproximability of rainbow coloring. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1479–1495. SIAM, 2020. doi:10.1137/1.9781611975994.90.

2 Per Austrin, Venkatesan Guruswami, and Johan Håstad. (2+ϵ)-SAT Is NP-hard. *SIAM Journal on Computing*, 46(5):1554–1573, 2017. doi:10.1137/15M1006507.

3 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *arXiv:1811.00970*, 2019. Version 3, 21 June 2019. arXiv:1811.00970.

4 Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei Krokhin and Stanislav Živný, editors, *Complexity and approximability of Constraint Satisfaction Problems*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/DFU.Vol7.15301.i.

5 Joshua Brakensiek and Venkatesan Guruswami. New Hardness Results for Graph and Hypergraph Colorings. In *Proceedings of the 31st Conference on Computational Complexity (CCC'16)*, volume 50 of *LIPIcs*, pages 14:1–14:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.14.

6 Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Structure Theory and a Symmetric Boolean Dichotomy. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 1782–1801. SIAM, 2018. doi:10.1137/1.9781611975031.117.

7 Joshua Brakensiek and Venkatesan Guruswami. An Algorithmic Blend of LPs and Ring Equations for Promise CSPs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pages 436–455. SIAM, 2019. doi:10.1137/1.9781611975482.28.

8 Joshua Brakensiek and Venkatesan Guruswami. Symmetric polymorphisms and efficient decidability of PCSPs. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 297–304. SIAM, 2020. doi:10.1137/1.9781611975994.18.

9 Alex Brandts, Marcin Wrochna, and Stanislav Živný. The complexity of promise SAT on non-Boolean domains. *CoRR*, abs/1911.09065, 2019. arXiv:1911.09065.

10 Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC'19)*, pages 602–613. ACM, 2019. doi:10.1145/3313276.3316300.

11 Hubie Chen and Martin Grohe. Constraint satisfaction with succinctly specified relations. *J. Comput. Syst. Sci.*, 76(8):847–860, 2010. doi:10.1016/j.jcss.2010.04.003.

12 Stephen Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC'71)*, pages 151–158, 1971. doi:10.1145/800157.805047.

13 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM J. Comput.*, 34(5):1129–1146, 2005. doi:10.1137/S0097539704443057.

14 Irit Dinur, Oded Regev, and Clifford D. Smyth. The hardness of 3-uniform hypergraph coloring. *Combinatorica*, 25(5):519–535, 2005. doi:10.1007/s00493-005-0032-4.

15 Miron Ficak, Marcin Kozik, Miroslav Olšák, and Szymon Stankiewicz. Dichotomy for Symmetric Boolean PCSPs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP'19)*, volume 132, pages 57:1–57:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.57.

16 M. R. Garey and David S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976. doi:10.1145/321921.321926.

17 Àngel J. Gil, Miki Hermann, Gernot Salzer, and Bruno Zanuttini. Efficient algorithms for description problems over finite totally ordered domains. *SIAM J. Comput.*, 38(3):922–945, 2008. doi:10.1137/050635900.

18 Venkatesan Guruswami and Sai Sandeep. Rainbow coloring hardness via low sensitivity polymorphisms. *SIAM J. Discrete Math.*, 34(1):520–537, 2020. doi:10.1137/19M127731X.

19 Subhash Khot. Hardness results for coloring 3-colorable 3-uniform hypergraphs. In *Proc. 43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 23–32. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181879.

20 Andrei Krokhin and Jakub Opršal. The complexity of 3-colouring H-colourable graphs. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS'19)*, pages 1227–1239. IEEE, 2019. doi:10.1109/FOCS.2019.00076.

21 Melven Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 13:15–20, 1967. doi:10.1002/malq.19670130104.

22 Leonid Levin. Universal search problems (in Russian). *Problems of Information Transmission (in Russian)*, 9(3):115–116, 1973. URL: http://www.mathnet.ru/links/84e4c96b64cc22a33a4bdae3d4815887/ppi914.pdf.

23 Christos H. Papadimitriou. On selecting a satisfying truth assignment. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'91)*, pages 163–169. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185365.