CNF Satisfiability in a Subspace and Related Problems

Vikraman Arvind ✉
The Institute of Mathematical Sciences (HBNI), Chennai, India

Venkatesan Guruswami ✉
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We introduce the problem of finding a satisfying assignment to a CNF formula that must further belong to a prescribed input subspace. Equivalent formulations of the problem include finding a point outside a union of subspaces (the Union-of-Subspace Avoidance (USA) problem), and finding a common zero of a system of polynomials over $\mathbb{F}_2$ each of which is a product of affine forms.

We focus on the case of k-CNF formulas (the k-Sub-Sat problem). Clearly, k-Sub-Sat is no easier than k-SAT, and might be harder. Indeed, via simple reductions we show that 2-Sub-Sat is NP-hard, and W[1]-hard when parameterized by the co-dimension of the subspace. We also prove that the optimization version Max-2-Sub-Sat is NP-hard to approximate better than the trivial 3/4 ratio even on satisfiable instances.

On the algorithmic front, we investigate fast exponential algorithms which give non-trivial savings over brute-force algorithms. We give a simple branching algorithm with running time $(1.5)^r$ for 2-Sub-Sat, where r is the subspace dimension, as well as an $O^*(1.4312)^n$ time algorithm where n is the number of variables.

Turning to k-Sub-Sat for $k \geqslant 3$, while known algorithms for solving a system of degree k polynomial equations already imply a solution with running time $\approx 2^{r(1-1/2k)}$, we explore a more combinatorial approach. Based on an analysis of critical variables (a key notion underlying the randomized k-SAT algorithm of Paturi, Pudlak, and Zane), we give an algorithm with running time $\approx \binom{n}{\leqslant t} 2^{n-n/k}$ where n is the number of variables and t is the co-dimension of the subspace. This improves upon the running time of the polynomial equations approach for small co-dimension. Our combinatorial approach also achieves polynomial space in contrast to the algebraic approach that uses exponential space. We also give a PPZ-style algorithm for k-Sub-Sat with running time $\approx 2^{n-n/2k}$. This algorithm is in fact oblivious to the structure of the subspace, and extends when the subspace-membership constraint is replaced by any constraint for which partial satisfying assignments can be efficiently completed to a full satisfying assignment. Finally, for systems of $O(n)$ polynomial equations in n variables over $\mathbb{F}_2$, we give a fast exponential algorithm when each polynomial has bounded degree irreducible factors (but can otherwise have large degree) using a degree reduction trick.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases CNF Satisfiability, Exact exponential algorithms, Hardness results

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.5

Related Version *Full Version*: https://arxiv.org/abs/2108.05914 [1]

Funding *Venkatesan Guruswami*: Portions of this work were done during visits to the Institute of Mathematical Sciences, Chennai. Research supported in part by the National Science Foundation grant CCF-1908125 and a Simons Investigator Award.

Acknowledgements We thank anonymous reviewers for useful comments and pointers to the literature.

16th International Symposium on Parameterized and Exact Computation (IPEC 2021).
Editors: Petr A. Golovach and Meirav Zehavi; Article No. 5; pp. 5:1–5:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Given an n-variate Boolean formula Φ along with an affine subspace $A \subseteq \mathbb{F}_2^n$ (given by a system of $\mathbb{F}_2$-linear equations) as input, we explore the complexity of testing if Φ has a satisfying assignment in A. This is a natural twist on Boolean constraint satisfaction problems that studies the effects of linear algebra on Boolean logic. Our focus shall be on the case when Φ is presented in Conjunctive Normal Formal (CNF). We refer to this problem as *satisfiability in a subspace* and denote it by SUB-SAT. This framework can capture non-Boolean problems such as Graph K-Colorability indicating the richness of combining the problem of Boolean CNF-satisfiability with a linear-algebraic constraint. We also note that in the area of practical SAT solvers there is interest in CNF satisfiability conjuncted with XOR constraints [26, 25].

Further, SUB-SAT has two other equivalent interesting formulations. The first of these is *union of subspace avoidance*, USA for short: Given affine subspaces $A_1, A_2, \ldots, A_m \subseteq \mathbb{F}_2^n$ is there an $x \in \mathbb{F}_2^n$ that is not in the union $\bigcup_{i=1}^m A_i$? A different formulation is a special case of finding a solution to a bunch of polynomial equations $p_i = 0$ over $\mathbb{F}_2^n$, namely when each p_i is a product of affine forms. We refer to this reformulation as PAF-SAT. We will describe these (easy) equivalences in Section 1.3.

For most of the paper, we restrict attention to the case when Φ is a k-CNF formula (a CNF formula with clauses of width at most k) for a fixed k, referred to as the k-SUB-SAT problem. Clearly, k-SUB-SAT is a generalization of the well-studied k-SAT (k-CNF satisfiability). In terms of the two reformulations above, k-SUB-SAT corresponds to the USA problem when the spaces A_i have co-dimension at most k, and for the PAF-SAT problem, each polynomial p_i is the product of up to k affine forms.

We present both hardness results and algorithms for k-SUB-SAT, described in Sections 1.1 and 1.2 below respectively. Owing to the NP-hardness of the problems, the algorithmic focus is on exponential time algorithms that give non-trivial improvements over brute-force.

There are two possible angles from which to view the study of k-SUB-SAT. The first is as a problem intermediate between satisfiability of k-CNF formula and a system of degree k polynomial equations. The second is as a specific instance of a constraint satisfaction problem (CSP) obtained by combining two fundamental types of constraints. There have been a few works [20, 7] giving algorithms beating brute-force for some natural problems with mixed constraints, but we are still far from a general picture of how to obtain fast exponential algorithms for a combined template of constraints when each constraint type does admit such non-trivial algorithms. In this context, tackling the combination of k-CNF formulas and linear equations is a good starting point, and one that could hopefully spur a more systematic study in the future. There have been a few investigations [15, 17, 8, 16] into the fine-grained complexity of CSPs via the algebraic approach based on (partial) polymorphisms. This theory has developed the tools to compare the optimal exponents of different constraint types, identifying for instance the "easiest" NP-hard CSP within some classes. However, with the exception of [4], polymorphisms have not been leveraged to design fast exponential algorithms with competitive exponents.

1.1 Hardness results

Since k-SUB-SAT is a generalization of k-SAT, k-SUB-SAT inherits all the intractability results of k-SAT for $k \geqslant 3$. This leaves the interesting case of 2-SUB-SAT. This turns out to be much harder than the polynomial time solvable 2-SAT. We establish the following, showing not just hardness (even for FPT algorithms) of the exact version, but also a tight inapproximability for the approximation version (even on satisfiable instances). The proofs are based on short, simple reductions, once an appropriate problem to reduce from is chosen.[1] The W[1]-hardness answers a question posed in [3] on the fixed-parameter complexity of 2-SAT with a global modular constraint, parameterized by the modulus.

▶ **Theorem 1.**

1. *2-SUB-SAT is* NP*-hard. It is further* W[1]*-hard when parameterized by the co-dimension of the affine space A in which we seek a satisfying assignment.*
2. *Given a satisfiable instance of 2-SUB-SAT, it is* NP*-hard to find an assignment in the input space A that satisfies more than $3/4 + \epsilon$ of the 2SAT clauses, for any $\epsilon > 0$.*

1.2 Algorithmic results

Analogous to seeking k-SAT algorithms faster than brute-force, we investigate fast exponential time algorithms for k-SUB-SAT that beat the naive brute-force $2^{\dim(A)}$ time algorithm, where $A \subseteq \mathbb{F}_2^n$ is the subspace in which we seek a solution. Algorithms for k-SAT have received much attention and are central to the burgeoning field of fast exponential-time algorithms. The algorithmic theory is closely connected to fixed parameter tractability and parameterized complexity [11, 10]. The accompanying hardness theory [13, 14], based on the exponential-time hypothesis (ETH) and the strong exponential-time hypothesis (SETH), is a sanity check to the quest for faster algorithms for k-SAT and other NP-complete problems.

There are several interesting k-SAT algorithms with running time $O^*(2^{n(1-\Theta(1/k))})$.[2] We only mention two significant algorithms from among these: one by Paturi, Pudlak, Zane [21] and another due to Schöning [23]. Both algorithms are simple to describe with delightfully clever and elegant analyses. The PPZ algorithm considers variables in a random order, and gives each a random value unless its value is forced by a clause and previously set values. It achieves a running time of $O^*(2^{n(1-1/k)})$. Schöning's algorithm starts with a random assignment and in each step fixes an unsatisfied clause by flipping the value of a random one of its variables. It achieves a running time of $O^*((2-2/k)^n)$.

Given that k-SUB-SAT generalizes k-SAT, it is natural to seek exponential algorithms with similar running times for k-SUB-SAT. For SUB-SAT with input space $A \subseteq \mathbb{F}_2^n$, the brute-force algorithm in fact runs in time $O^*(2^{\dim(A)})$. A natural question is whether we can get similar improvements in the exponent of the $O^*(2^{\dim(A)})$ running time.

An algorithm [18] with running time about $O^*(2^{r(1-1/5k)})$ is known for checking satisfiability of a collection of arbitrary degree k polynomial equations in r variables: Let $P_i \in \mathbb{F}_2[x_1, x_2, \ldots, x_r]$, $1 \leqslant i \leqslant m$, be polynomials over the field $\mathbb{F}_2$. Following [18], the POLY-EQS problem is solving the system of polynomial equations $P_i = 0, 1 \leqslant i \leqslant m$ over $\mathbb{F}_2$: to check if there exists a solution in $\mathbb{F}_2^r$ and compute one if it exists. When P_i are all of degree bounded by k we denote this special case by k-POLY-EQS. The k-POLY-EQS problem generalizes k-SUB-SAT by the following easy transformation: Suppose the subspace A where we seek a satisfying assignment is r dimensional. Then we can express the i^{th} clause in the k-SUB-SAT instance as a disjunction of k affine linear forms in r variables: $C_i = (\ell_{i,1} \vee \ell_{i,2} \vee \cdots \vee \ell_{i,k})$. We define the corresponding polynomial $P_i = \prod_{j=1}^{k}(\ell_{i,j} + 1)$. Now, the k-SUB-SAT instance is satisfiable iff the k-POLY-EQS instance $P_i = 0, 1 \leqslant i \leqslant m$ has a solution in $\mathbb{F}_2^r$.

[1] The NP-hardness would also follow from Schaefer's dichotomy theorem for Boolean CSP [22], though that is an overkill hammer for this result.

[2] The notation $O^*(f(n))$ for running time bounds suppresses polynomial factors.

The algorithm [18] is a novel application of the Razborov-Smolensky "polynomial method," originally developed as a lower bound technique, used to define low-degree probabilistic polynomials for approximating the OR gate. The same idea allows for replacing a system of polynomial equations by a single probabilistic polynomial (without significant increase in degree), followed by a partial table lookup search. The article [18] presents more general results applicable to all finite fields $\mathbb{F}_q$. Recently, in [9], the running time for the case of $\mathbb{F}_2$ has been improved to $O^*(2^{r(1-1/2k)})$ by a refinement of the search method in [18].

Since k-Sub-Sat is a special case of solving a system of polynomial equations over $\mathbb{F}_2$, it raises the natural question of improving the running time further to match the $O^*(2^{r(1-1/k)})$ running time of the PPZ randomized algorithm for k-Sat. We are only able to achieve this speed-up in some special cases. However, on the positive side, our algorithms turn out to be *polynomial space bounded*, unlike the polynomial equations based method which requires exponential space [18, 9].

1.2.1 Algorithms for 2-Sub-Sat

For 2-Sub-Sat a simple deterministic branch-and-bound algorithm achieves a running time of $O^*(3^{r/2})$ where r is the dimension of the subspace A. We can improve on this with a randomized branching strategy to a running time of $O^*(1.5^r)$. This improves over the randomized $O^*(1.6181^r)$ algorithm given by the polynomial method [9] for solving a system of quadratic equations over $\mathbb{F}_2$. There is also a simple deterministic branching algorithm with $O^*(((1+\sqrt{5})/2)^r)$ running time for 2-Sub-Sat. This is based on the same branching strategy for k-Sat [19, Theorem, pp. 295] with its running time governed by the generalized Fibinacci numbers.

When $\dim(A) = n - t$, we can adapt the algorithm from [3, Algorithm 4.1] (for solving 2-SAT with a single abelian group constraint) to obtain an $O^*(\binom{n}{\leqslant t})$ time algorithm. [3]

The result of Theorem 1 shows that this problem is not in FPT parameterized by the co-dimension t, answering a question posed in [3] on whether 2-SAT with a global abelian group constraint might be fixed-parameter tractable, parameterized by the group size. More generally, the work [3] systematically studied the effect of a global modular constraint on the complexity of Boolean constraint satisfaction problems, exposing many interesting phenomena and connections.

Balancing the two running times of $O^*(1.5^r)$ and $O^*(\binom{n}{n-r})$ algorithm when $r \geqslant n/2$ (the exponents of the two bounds become equal at $r = (1-\eta)n$ for $\eta \approx 0.115816$) yields a $O^*(1.4312^n)$ time randomized algorithm for 2-Sub-Sat on n variables. The following records these results.

▶ **Theorem 2.** *There is a randomized $O^*(1.5^r)$ algorithm for 2-Sub-Sat where r is the dimension of the input space, as well a deterministic $O^*(\binom{n}{\leqslant t})$ time algorithm where t is the co-dimension. Together, these imply a randomized $O^*(1.4312^n)$ time algorithm as a function of the number n of variables.*

1.2.2 Algorithms for k-Sub-Sat

We explore combinatorial algorithms for k-Sub-Sat based on the notion of *critical variables* (which was introduced in [21] and plays an important role in their satisfiability algorithm). Let Φ be a satisfiable CNF formula in n variables $x_i, i \in [n]$, and let $\bar{a} \in \mathbb{F}_2^n$ be a satisfying assignment.

---

[3] For nonnegative integers n, t, the notation $\binom{n}{\leqslant t}$ stands for $\sum_{i=0}^{t} \binom{n}{i}$.

▶ **Definition 3** ([21]). *We say x_i is a* critical variable *for $\bar{a}$ with respect to Φ if the assignment $\bar{a} + e_i$ falsifies Φ, where e_i is the i^{th} elementary vector with 1 in the i^{th} coordinate and zero elsewhere (so $\bar{a} + e_i$ is just $\bar{a}$ with x_i flipped). If the formula Φ is clear from context, we simply say that x_i is a critical variable for assignment $\bar{a}$.*

The key idea in our combinatorial algorithms is *plucking* of non-critical variables based on the following simple observation: if Φ is an n-variate CNF formula and $\bar{a}$ is a satisfying assignment such that variable x_i is non-critical for it, then the formula Φ' obtained by plucking x_i (i.e., dropping all occurrences of x_i and its complement from Φ) remains satisfiable with $\bar{a}' \in \mathbb{F}_2^{n-1}$ as a satisfying assignment, where $\bar{a}'$ is obtained from $\bar{a}$ by dropping the i^{th} coordinate.

The important property of Φ' is that given any satisfying assignment for Φ' we can set x_i to either 0 or 1 to recover a satisfying assignment for Φ. This facilitates searching for a satisfying assignment in an affine space A: if the plucked variable x_i occurs in a linear constraint defining A then we can drop that linear constraint while seeking a satisfying assignment for Φ', because that linear constraint can always be satisfied by choosing the right value of x_i which still remains overall a satisfying assignment for Φ. Based on this idea we obtain the following algorithms for k-SUB-SAT:

- The first result here is a randomized $O^*(\binom{n}{t}2^{n-n/k})$ time algorithm for k-SUB-SAT where $t = \mathrm{codim}(A)$. This algorithm is essentially governed by the running time of the PPZ satisfiability algorithm [21] combined with an iterative "search and pluck" operation to remove t non-critical variables from the t linear equations defining A. This running time is superior to the $O^*(2^{r-r/2k})$ time randomized algorithm based on solving polynomial equations for small values of $t = o(n)$.
- The second result is a general randomized $O^*(2^{n-n/2k+n/2k^2})$ time algorithm for k-SUB-SAT, nearly matching the $\approx 2^{r-r/2k}$ run time of the polynomial equations algorithm [9, 18] for r close to n. It again uses the PPZ satisfiability algorithm as a subroutine combined with simple applications of the plucking step: if the number of critical variables is fewer than $n/2$, it randomly guesses and plucks non-critical variables. This algorithm does not need to look at the linear equations defining A. In fact, it works for any Boolean constraint $C(x_1, x_2, \ldots, x_n)$ (replacing membership in the affine space A) with a polynomial-time algorithm that takes a partial assignment and extends it to an assignment that satisfies C. For example, C can be a HORN or dual HORN formula.
- It is pleasing to note that we can apply the idea of plucking non-critical variables to 2-SUB-SAT and obtain an $O^*(\binom{n}{\leqslant t})$ deterministic algorithm (cf. [3]), where $t = \mathrm{codim}(A)$. Exploiting the structure of 2-CNF formulas, we can find the non-critical variables efficiently.

▶ **Theorem 4.** *The k-SUB-SAT problem admits two randomized algorithms, one running in time $O^*(2^{n-n/2k+n/2k^2})$, and another running in $O^*(\binom{n}{t}2^{n-n/k})$ when the input subspace has co-dimension $t \leqslant n/2$.[4] Both algorithms use space bounded by a polynomial in n.*

▶ Remark 5. Satisfiability algorithms based on the switching lemma (which converts k-CNF to decision trees of moderate term size and number of terms) are known in the literature (e.g., see [12]). We can easily adapt this algorithm to solve k-SUB-SAT, because once we have a decision tree for the underlying k-CNF formula, for the k-SUB-SAT instance each path of the decision tree will give rise to a system of linear equations over $\mathbb{F}_2$. For each path, therefore,

[4] Of course, there is also a trivial $O^*(2^{n-t})$ time brute force algorithm.

we can even count the number of satisfying assignments. Counting over all the paths of the decision tree gives the total number of satisfying assignments for the k-Sub-Sat instance in randomized time $O^*(2^{n(1-1/c\cdot k)})$ for some suitable large constant $c > 0$. Furthermore, the algorithm is also polynomial space-bounded. In terms of running time, however, it is a much weaker bound in comparison to [18] or even the algorithms of Theorem 4. In this context, we note that for $\#k$-Sat there is a *deterministic* $O^*(2^{n(1-1/c\cdot k)})$ time algorithm based on the polynomial method (albeit using exponential space) [6]. We do not know of any such deterministic algorithm for counting satisfying assignments to k-Sub-Sat.

Finally, motivated by the (unbounded CNF) Sub-Sat problem, we revisit the general problem solving a system of polynomial equations $p_i = 0, 1 \leqslant i \leqslant m$ over $\mathbb{F}_2$, where $m = O(n)$, where each p_i is given by an arithmetic circuit of poly(n) degree. In the case when each p_i has small degree irreducible factors, we get a $2^{r(1-\alpha)}$ time randomized algorithm, where α depends on the number of equations m and the degree bound on the irreducible factors (Theorem 25).

1.3 Equivalent and related problems to Sub-Sat

Recall the USA problem: Given a collection of affine subspaces $A_1, A_2, \ldots, A_m \subseteq \mathbb{F}_2^n$ (where each A_i is given by a bunch of affine linear equations over $\mathbb{F}_2$) the problem is to determine if there is a point $x \in \mathbb{F}_2^n \setminus \bigcup_{i=1}^m A_i$.

Clearly, the complement $\mathbb{F}_2^n \setminus \bigcup_{i=1}^m A_i$ is expressible as an AND of ORs of affine linear forms $\oplus_{i\in S} x_i + b$, $b \in \{0,1\}$. Thus, USA is clearly reducible to Sub-Sat. The converse reduction is also easy: given a CNF formula Φ and an affine subspace $A \subseteq \mathbb{F}_2^n$ we first convert it to an AND of ORs of affine linear forms. An assignment $x \in A$ satisfies Φ if and only if it satisfies $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each clause C_i is an OR of affine linear forms. The set A_i of satisfying assignments of the complement $\overline{C_i}$ is an affine subspace of $\mathbb{F}_2^n$, and Φ is satisfiable by $x \in A$ if and only if $x \in \mathbb{F}_2^n \setminus \bigcup_{i=1}^m A_i$.

For the equivalence to PAF-Sat, suppose $\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each clause C_i is an OR of affine linear forms $C_i = \vee_{j=1}^t L_{ij}$. As already discussed in Section 1.2, the assignment $x \in \mathbb{F}_2^n$ satisfies C_i if and only if it satisfies the polynomial equation $\prod_{j=1}^m (L_{ij} + 1) = 0$. Thus, the satisfiability of Φ is reducible to a system of m polynomial equations $p_i = 0$, where each p_i is a product of affine linear forms. The converse reduction is also easy which we omit.

Organization of the paper

We present the results in a different order than in the introduction. In Section 2 we first present the algorithms for k-Sub-Sat and then for 2-Sub-Sat. In Section 3 we present our hardness results for 2-Sub-Sat. Finally, in Section 4 we present the algorithm for Poly-Eqs for $O(n)$ equations $p_i = 0$, where each p_i has unrestricted degree but constant-degree irreducible factors.

For reasons of space, all proofs are skipped in the extended abstract; a full version of the paper is available on arXiv [1].

2 Algorithmic results for k-Sub-Sat

As mentioned in the introduction, the k-Sub-Sat problem seems intermediate in difficulty, between k-Sat and the problem k-Poly-Eqs of solving a system of degree-k polynomial equations over $\mathbb{F}_2$. The latter problem has an $O^*(2^{r(1-1/2k)})$ time algorithm [18, 2, 9], which yields an $O^*(2^{r(1-1/2k)})$ time algorithm for k-Sub-Sat, where $r = \dim(A)$.

Ideally, we would like an algorithm for k-Sub-Sat with run time $O^*(2^{r(1-1/k)})$, with savings in the exponent similar to that of the PPZ algorithm [21] for k-Sat.

We present some algorithms in this direction: For 2-Sub-Sat there is a simple $O^*(1.5^r)$ time randomized algorithm which improves on the $O^*(2^{r(1-1/2k)})$ bound for $k = 2$. For a special case of k-Sub-Sat, when $r = \dim(A)$ is close to the number of variables n, we are able to adapt the PPZ algorithm to essentially get an $O^*(2^{r(1-1/2k)})$ time algorithm. Writing $t = n - r = \text{codim}(A)$, we can even obtain an $O^*(\binom{n}{\leqslant t} \cdot 2^{n(1-1/k)})$ time algorithm for the problem, also based on the PPZ satisfiability algorithm, which yields the desired $1/k$ savings in the exponent for small t.

2.1 An $O^*(\binom{n}{t} \cdot 2^{n(1-1/k)})$ time randomized algorithm: co-dimension t case

As outlined in Section 1.2, the algorithm will use the PPZ satisfiability algorithm [21] as a subroutine, combined with variable plucking steps to solve k-Sub-Sat in randomized time $O^*(\binom{n}{t} \cdot 2^{n(1-1/k)})$, when $\text{codim}(A) = t$. In particular, for $\text{codim}(A) = o(n)$ the algorithm has run time $O^*(2^{n(1-1/k+o(1))})$.

The variable plucking is based on analyzing the critical variables for a solution $\bar{a} \in \mathbb{F}_2^n$ of a given k-Sub-Sat instance (Φ, A), depending on whether or not they occur in the linear equations defining A.

For an instance (Φ, A) we partition the variables into two sets

$$\{x_i \mid i \in [n]\} = V_{\text{in}} \sqcup V_{\text{out}},$$

where V_{in} is the subset of variables that have nonzero coefficient in at least one of the t linear equations defining A, and V_{out} is the remaining set of variables. By abuse of notation, we will also treat $V_{\text{in}} \sqcup V_{\text{out}}$ as a partition of the index set $[n]$. We consider the following two cases.

Case 1. Suppose (Φ, A) has the property that *for every solution* $\bar{a} \in \mathbb{F}_2^n$ each variable in V_{in} is critical for $\bar{a}$ w.r.t Φ. There is no variable plucking required in this case. It only involves the application of the PPZ satisfiability algorithm on Φ and checking that the assignment found belongs to A. We need the following lemma which is analogous to [21, Lemma 4]. The proof of the lemma is by an induction argument like in [21].

▶ **Lemma 6.** *Let S be a nonempty subset of $\mathbb{F}_2^n$. For each $\bar{a} \in S$, let $I_{out}(\bar{a}) = \{i \in V_{out} \mid \bar{a} + e_i \notin S\}$, where e_i is the i^{th} elementary vector. Then we have*

$$\sum_{\bar{a} \in S} 2^{|I_{out}(\bar{a})| - |V_{out}|} \geqslant 1. \tag{1}$$

Now, let $\bar{a} \in \mathbb{F}_2^n$ be some solution of the k-Sub-Sat instance (Φ, A). Then, by the assumption of Case 1 and the preceding discussion $\bar{a}$ has $|V_{\text{in}}| + |I_{\text{out}}(\bar{a})|$ critical variables w.r.t Φ.

Following the analysis in [21], if we now run one iteration of the PPZ algorithm on the instance Φ, the probability that $\bar{a}$ is output is at least

$$\frac{1}{n^2} \cdot 2^{-n + (|V_{\text{in}}| + |I_{\text{out}}(\bar{a})|)/k}.$$

Let $S \subset \mathbb{F}_2^n$ denote the subset of solutions to the instance (Φ, A). Summing up over all $\bar{a} \in S$, the probability that some solution $\bar{a}$ is output is given by

$$\sum_{\bar{a}\in S} \frac{1}{n^2} \cdot 2^{-n+(|V_{\text{in}}|/k+|I_{\text{out}}(\bar{a})|/k)} = \frac{1}{n^2} 2^{-n+n/k} \cdot \sum_{\bar{a}\in S} 2^{(-|V_{\text{out}}|/k+|I_{\text{out}}(\bar{a})|)/k}$$
$$\geqslant \frac{1}{n^2} 2^{-n+n/k} \cdot \sum_{\bar{a}\in S} 2^{(-|V_{\text{out}}|+|I_{\text{out}}(\bar{a})|)} \geqslant \frac{1}{n^2} 2^{-n+n/k} ,$$

where the last step uses Lemma 6. This finishes the analysis of Case 1.

▶ Remark 7. Notice in the probability analysis that S is the set of solutions to (Φ, A) and not all solutions to Φ. The crucial property that for every $\bar{a} \in S$, each variable in V_{in} is critical w.r.t Φ yields that there are $|V_{\text{in}}| + |I_{\text{out}}(\bar{a})|$ critical variables for $\bar{a}$ w.r.t Φ. Intuitively, as the variables in V_{out} do not occur in the linear equations, the PPZ algorithm when run on Φ will be able to deterministically set, on average, $|I_{\text{out}}(\bar{a})|/k$ many of the critical variables in V_{out} without any interaction with the linear equations defining A.

Case 2. We now consider the case when not all variables in V_{in} are critical to all solutions to (Φ, A). We will show that there is a subset of at most t variables in V_{in} that can be plucked from Φ and reduce the transformed instance to Case 1. We will argue that the algorithm can do an exhaustive search for this subset of V_{in} of size at most t.

▶ **Lemma 8.** *In the k-SUB-SAT instance (Φ, A), let $Bx = b$ be the system of t linear equations defining A. Suppose variable x_1 occurs in the first equation $\sum_{j=1}^n B_{1j}x_j = b_1$ (i.e., $B_{11} \neq 0$). Further, suppose x_1 is not critical for some solution to (Φ, A). Let Φ' be the formula obtained by plucking x_1 from Φ. Let A' be the affine space of co-dimension $t-1$ defined by dropping the first linear equation $\sum_{j=1}^n B_{1j}x_j = b_1$* after eliminating x_1 *from the other linear equations by row operations. Then (Φ', A') is satisfiable and* any solution $\bar{a'}$ *to (Φ', A') can be extended to a solution $\bar{a}$ of (Φ, A).*

Lemma 8 describes a pluck/eliminate step applied to the non-critical variable x_1: namely, pluck x_1 from Φ and eliminate it from the equations describing A.

Clearly, for some sequence of $s \leqslant t$ pluck/eliminate steps applied successively transforms (Φ, A) to (Φ_s, A_s) for which Case 1 holds. Since we do not have an efficient test for checking non-criticality, the algorithm has to do an exhaustive search for the sequence of s variables to pluck/eliminate. The number of variable sequences to consider is bounded by n^t. However, as we argue in the next claim, it suffices to consider each unordered subset U of size $s \leqslant t$ variables and apply pluck/eliminate steps to its variables in the natural order $x_1, \ldots, x_n$. Thus, we can bound the exhaustive search to $\binom{n}{\leqslant t}$ subsets of variables. Let (Φ_U, A_U) be the resulting instance after pluck/eliminate applied to variables in U in the natural order.

▶ **Lemma 9.** *Let (Φ, A) be a satisfiable instance of k-SUB-SAT with* $\text{codim}(A) = t$. *There is a subset U of variables of size at most t, such that (Φ_U, A_U) is a satisfiable Case 1 instance of k-SUB-SAT.*

The $O^*(\binom{n}{\leqslant t} \cdot 2^{n-n/k})$ time Algorithm. On input (Φ, A), the algorithm proceeds as follows:

For each subset $U \subset V_{\text{in}}$ of size at most t do the following:
1. Pluck the variables in U from Φ to obtain Φ_U.

2. For each variable $x_i \in U$ (in any order): pick some equation in which x_i occurs; remove x_i from other equations by adding the picked equation to it; drop the picked equation from the system.
3. Run the PPZ algorithm on the resulting instance (Φ_U, A_U) as if Case 1 were applicable. More precisely, run PPZ on Φ_U for $O^*(2^{n-n/k})$ steps; for each solution obtained, if it satisfies A_U then output an extension of it to a solution to (Φ, A) and exit,[5] else continue the for-loop for the next choice of subset U.

To see the correctness, suppose (Φ, A) is satisfiable. By Lemma 9, for some choice of U with $|U| \leqslant t$, (Φ_U, A_U) is a Case 1 instance. Hence, the PPZ satisfiability algorithm will output a solution to (Φ_U, A_U) in time $O^*(2^{n-n/k})$ with high probability. This solution can be uniquely extended to a solution to (Φ, A) using the linear equations.

We have thus shown the following.

▶ **Theorem 10.** *There is a randomized $O^*(\binom{n}{t} \cdot 2^{n-n/k})$ time algorithm for k-Sub-Sat for subspaces of co-dimension t. In particular, for $t = o(n)$ we have a randomized $O^*(2^{n(1-1/k+o(1))})$ time algorithm.*

2.2 An $O^*(2^{n-n/2k+n/2k^2})$ time PPZ-based algorithm for k-Sub-Sat

Let (Φ, A) be a k-Sub-Sat instance. Our objective is a randomized algorithm with run time $2^{n-(1-\nu)n/k}$ for as small an ν as possible (ideally, tending to zero).

To this end, we can first apply Valiant-Vazirani Lemma [27] to increase the number of constraints (thereby reducing the rank of A) and getting an instance (Φ, A') such that Φ has a unique solution in A' with high probability (i.e., inverse polynomial probability as guaranteed by Valiant-Vazirani).

If $\dim(A') \leqslant n - (1-\nu)n/k$ we can brute force search in A' in deterministic time $2^{\dim(A')} \leqslant 2^{n-(1-\nu)n/k}$. Thus, we can assume that $\dim(A') = n - t$ and A' is the solution space of $t < (1-\nu)n/k$ independent affine linear equations.

Let now $\bar{a} \in \mathbb{F}_2^n$ be the unique solution to the k-Sub-Sat instance (Φ, A'). We partition the variable set into $V_{\text{in}} \sqcup V_{\text{out}}$ as before.

▷ Claim 11. Every variable in V_{out} is critical for the satisfying assignment $\bar{a}$ of Φ.

Proof. Suppose $x_i \in V_{\text{out}}$ is not critical for $\bar{a}$. Then $\bar{a} + e_i$ is also a satisfying assignment for Φ. Moreover, since x_i does not occur in V_{in}, $\bar{a} + e_i$ satisfies the linear equations defining A'. Hence $\bar{a} + e_i$ is a solution to (Φ, A') contradicting the uniqueness of $\bar{a}$.

The variable plucking algorithm. If $\bar{a}$ has more than $(1-\nu)n$ many critical variables (ν to be fixed in the analysis) then by running the PPZ satisfiability algorithm [21] for $O^*(2^{n-(1-\nu)n/k})$ iterations we will find it with high probability.

Otherwise, there are more than νn many variables in V_{in} that are *not critical* for Φ at $\bar{a}$.

1. Repeat the following two steps at most t times.
2. (The plucking step) Randomly pluck a variable x_i from V_{in} and drop it from the formula Φ to obtain its *shrinking* Φ_1. Take a linear equation $\ell = b$ in which x_i occurs. By row operations eliminate x_i from all other linear equations in which x_i occurs and then drop the equation $\ell = b$. Let the affine space described by the new set of at most $t-1$ linear

[5] From a solution to (Φ_U, A_U) we can reconstruct the solution to (Φ, A) as the values to variables in U are uniquely determined via the linear equations from the values to the other variables.

equations be A_1. We claim that (Φ_1, A_1) also has a unique solution $\bar{a}_1$ (obtained from $\bar{a}$ by dropping the i^{th} coordinate).

3. Let $n_1 = n - 1$. Run the PPZ algorithm for $2^{n_1-(1-\nu)n_1/k}$ time on Φ_1. If we do not find the unique solution $\bar{a}_1$ then repeat the plucking step.

At the end of t successful plucking steps we are left with a k-Sat instance Φ_t with a unique solution (the subspace A_t is $\mathbb{F}_2^n$) and PPZ will find that solution from which we can compute $\bar{a}$ by recovering the unique values of the plucked variables using the linear equations.

Analysis. At the j^{th} iteration of the plucking step, the probability that all j steps pluck off non-critical variables is at least ν^j. Thus, the running time of the search for unique solutions for the (Φ_j, A_j) over all t steps is bounded by $\sum_{j=0}^{t} O^*(\frac{1}{\nu^j} \cdot 2^{n_j-(1-\nu)n_j/k})$.

Letting $\alpha = 2^{1-(1-\nu)/k}$ and noting that $n_j = n - j$ we can rewrite and bound the above sum as

$$O^*(2^{n-(1-\nu)n/k}) \cdot \sum_{j=0}^{t} \frac{1}{\nu^j \cdot \alpha^j} \leqslant O^*(2^{n-(1-\nu)n/k}) \cdot t \cdot \frac{1}{\nu^t \cdot \alpha^t}$$

$$\leqslant O^*(2^{n-(1-\nu)n/k}) \cdot t \cdot \left(\frac{1}{2\nu}\right)^{(1-\nu)n/k} \cdot 2^{(1-\nu)n/k^2},$$

as the sum $\sum_{j=0}^{t} \frac{1}{\nu^j \cdot \alpha^j}$ is bounded by $t\frac{1}{\nu^t \cdot \alpha^t}$ for $\nu\alpha < 1$ and $t \leqslant (1-\nu)n/k$.

The overall running time of the algorithm is, therefore, $O^*(2^{n-n/k}) \cdot 2^{\nu n/k} \cdot \left(\frac{1}{2\nu}\right)^{(1-\nu)n/k} \cdot 2^{(1-\nu)n/k^2}$, which is minimized at $\nu = 1/2$ as we argue below, and is given by $O^*(2^{n-n/2k+n/2k^2})$.

▶ Remark 12 (Extension beyond linear-algebraic constraints). We note some aspects about the algorithm and explain its adaptation to the more general setting of k-CNF satisfiability in the presence of a global boolean constraint $C(x_1, x_2, \ldots, x_n)$ with the property that given a partial assignment to the variables x_i we can extend the assignment to the remaining variables that satisfies the constraint C, if such an extension exists. We set $\nu = 1/2$ and $t = n/2k$. Note that the algorithm need not partition the variables into V_{in} and V_{out}. If there are over $n/2$ non-critical variables, the algorithm can "obliviously" pluck one with probability 1/2. Oblivious in the sense that it does not need to see the constraint C. After $t = n/2k$ plucking steps, there are at most $n - n/2k$ remaining variables. We add a final step to the algorithm which is a brute-force search over all $2^{n-n/2k}$ assignments to the remaining variables. For each assignment to these that satisfies Φ_t we can check, in polynomial time, if there is an extension to it that satisfies C. This search will succeed for the unique solution $\bar{a}$. An interesting example for constraint C would be HORN formulas. As clause size is unrestricted in HORN formulas, notice that neither a direct application of the PPZ satisfiability algorithm, nor an application of the polynomial equations algorithms would give constant savings in the exponent for the running time bound.

More generally, call a Boolean constraint $C(x_1, x_2, \ldots, x_n)$ $T(n)$-easy if there is a $T(n)$ time-bounded algorithm that searches for a satisfying extension of a given partial assignment to the variables x_i.

▶ **Theorem 13.** *There is a randomized $O^*(2^{n-n/2k+n/2k^2} \cdot T(n))$ time algorithm that takes any k-CNF formula and a $T(n)$-easy boolean constraint $C(x_1, x_2, \ldots, x_n)$ as input and computes a satisfying assignment for the formula and C.*

▶ **Corollary 14.** *There is a randomized $O^*(2^{n-n/2k+n/2k^2})$ time algorithm for k-Sub-Sat.*

2.3 An $O^*(1.5^r)$ time algorithm for 2-Sub-Sat

▶ **Theorem 15.** *Given a* 2-SUB-SAT *instance* (Φ, A)*, where* Φ *is a* 2-CNF *formula and* $A \subset \mathbb{F}_2^n$ *is an* r*-dimensional affine subspace given by linear equations, there is a randomized* $O^*(1.5^r)$ *time algorithm to check if* Φ *has a satisfying assignment in* A *and if so to compute it.*

▶ Remark 16. The run time of $O^*(1.5^r)$ that we obtain improves on the polynomial equations based algorithms, where for $k = 2$ the best run time so far is $O^*(1.618^r)$ [9]. For $k = 3$ a similar randomized branching strategy gives an algorithm with run time $O^*((7/4)^r)$. For larger k the run time degrades to $O^*((2 - 1/2^{k-1})^r)$. This running time bound is obtained similarly as for Theorem 15: fix a satisfying assignment $\bar{a}$ of the k-SUB-SAT instance. For a clause $(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k)$ of k linearly independent linear forms a random (nonzero) linear combination $\sum_{i=1}^k \alpha_i \ell_i$ evaluates to 1 at $\bar{a}$ with probability exactly $\frac{2^{k-1}}{2^k-1}$.

2.4 2-Sub-Sat in a co-dimension t subspace

In this section we consider 2-SUB-SAT where we are seeking a solution in an affine space A such that $\mathrm{codim}(A) = t$.

Given a formula Φ we will identify a canonical satisfying assignment $\bar{a}$ for Φ based on which we will define critical variables. Since 2-SAT is in polynomial-time, we can detect non-critical variables in Φ w.r.t. $\bar{a}$ in polynomial time. Now the plucking step will try all the possible $\binom{n}{t}$ choices of plucking non-critical variables, recalling that a non-critical variable plucked from a linear constraint defining A allows us to drop that constraint.

▶ **Theorem 17.** *There is an* $O^*(\binom{n}{t})$ *time deterministic algorithm for checking if a* 2-SUB-SAT *instance* (Φ, A) *is satisfiable where the affine space* A *has co-dimension* t*.*

3 Hardness results

In this section we prove our hardness results for subspace satisfiability. Since k-SAT itself is NP-hard for $k \geqslant 3$, so is k-SUB-SAT for $k \geqslant 3$. So we focus on the case $k = 2$.

3.1 NP-hardness of 2-Sub-Sat

While 2-SAT is polynomial time solvable, the following theorem shows that 2-SUB-SAT is NP-hard. Note that this follows from Schaefer's dichotomy theorem for Boolean CSP as the combination of 2-SAT constraints and linear equations (even with 3 variables per equation) is not one of the six tractable cases, and thus NP-hard. Below we give a direct proof based on a simple reduction.

▶ **Theorem 18.** 2-*SUB-SAT is NP-hard.*

3.2 W[1]-hardness of 2-Sub-Sat parameterized by co-dimension

We now strengthen the hardness result of Theorem 18 and show that 2-SUB-SAT is unlikely to even be fixed-parameter tractable when parameterized by the co-dimension t of the subspace in which we seek a satisfying assignment to the 2CNF formula. On the other hand, recall that (as shown in [3] and also Section 2.4), for fixed co-dimension t, 2-SUB-SAT can be solved in polynomial time. Our W[1]-hardness answers (in the negative) a question posed in [3] on whether 2-SAT with a single modular constraint modulo M is fixed-parameter tractable when parameterized by M (they gave an algorithm with complexity $n^{O(M)}$).

▶ **Theorem 19.** *Consider the* 2-Sub-Sat *where the input subspace within which one has to satisfy the* 2-Sat *formula has co-dimension* t*. Parameterized by* t*,* 2-Sub-Sat *is* W[1]*-hard.*

3.3 Approximability of Max-2-Sub-Sat

Given the hardness of deciding exact satisfiability of 2-Sub-Sat instance, we now turn to approximate satisfiability. In the Max-2-Sub-Sat problem, the goal is to satisfy the maximum number of 2SAT clauses with an assignment that belongs to the input affine space A. Thus, the affine constraints are treated as hard constraints. We allow clauses of width 1. If unary clauses are disallowed in the 2CNF formula, and each clause involves exactly two distinct variables, we call the problem Max-E2-Sub-Sat.

3.3.1 Easy approximation algorithms

We can assume that no variable is forced to 0 or 1 by the affine space A, since if that happens we can just set and remove that variable and work on the reduced instance. If we pick a random assignment from A, it will satisfy at least $1/2$ of the clauses of the 2CNF formula in expectation, and in fact at least an expected fraction $3/4$ of the clauses when each clause involves two distinct variables. The algorithms are easily derandomized. For satisfiable instances of Max-2-Sub-Sat, one can find a $3/4$ approximate solution, as one can eliminate all the unary clauses, and add those conditions to the subspace inside which we want to find an assignment to the 2CNF formula. So we get the following trivial algorithmic guarantees.

▶ **Observation 20.** *In polynomial time, one can get a factor* $1/2$ *approximate solution to instances of* Max-2-Sub-Sat*, a factor* $3/4$ *approximate solution to instances of* Max-E2-Sub-Sat*, and a factor* $3/4$ *approximate solution to satisfiable instances of* Max-2-Sub-Sat*.*

We will now show that all the above guarantees are best possible, with matching NP-hardness results.

3.3.2 Tight inapproximability via simple reductions

For the hardness results and rest of the section, it is convenient to work with the PAF-Sat formulation of Sub-Sat. The Max-LIN2 problem, of maximizing the number of satisfied equations in a system of affine equations mod 2, trivially reduces to Max-2-PAF-Sat (with each equation being degree 1 instead of degree 2). By Håstad's seminal tight inapproximability for Max-LIN2, we have the following.

▶ **Observation 21.** *For any* $\epsilon > 0$*,* Max-2-PAF-Sat *(and thus* Max-2-Sub-Sat*) is NP-hard to approximate within a factor of* $(1/2 + \epsilon)$*, and this holds for almost satisfiable instances that admit an assignment satisfying a fraction* $(1 - \epsilon)$ *of equations.*

We also get a tight hardness (matching Observation 20) for the Max-E2-Sub-Sat or equivalently when each polynomial equation is the product of exactly two (linearly independent) affine forms.

▶ **Lemma 22.** *For any* $\epsilon > 0$*,* Max-E2-PAF-Sat *is NP-hard to approximate within a factor of* $(3/4+\epsilon)$*, and this holds for almost satisfiable instances that admit an assignment satisfying a fraction* $(1 - \epsilon)$ *of equations.*

3.3.3 Inapproximability for satisfiable instances

The above inpproximability results do not apply to *satisfiable* instances of 2-Sub-Sat. They are obtained by reductions from linear equations whose exact satisfiability can be easily checked. We now prove that approximating Max-2-Sub-Sat doesn't get easier on satisfiable instances.

▶ **Theorem 23.** *For every $\epsilon > 0$, it is NP-hard to approximately solve satisfiable instance of Max-E2-Sub-Sat within a factor of $3/4 + \epsilon$. That is, it is NP-hard to find, given as input a satisfiable instance of 2-Sub-Sat, an assignment satisfying a fraction $3/4 + \epsilon$ of the 2SAT constraints.*

4 System of polynomial equations over binary field: effect of reducibility

We now examine a special case of the problem of solving a system of polynomial equations over $\mathbb{F}_2$ studied in [18, 2, 9]. For motivating background, we recall according to the strong exponential time hypothesis (SETH) that Sat, that is n-variable CNF satisfiability of unrestricted clause width, cannot be essentially solved faster than 2^n time. However, Schuler [24] and Calabro et al [5] have shown the special case that sparse instances of Sat (with $c \cdot n$ clauses) can be solved in $O^*(2^{n(1-\alpha)})$ time, where α is a constant depending on the clause density c. It is natural to ask if there is an analogous result for Sub-Sat (satisfiability of conjunctions of unbounded disjunctions of affine linear forms). In this section we show a more general algorithmic result in the setting of systems of polynomial equations over $\mathbb{F}_2$.

Let $P_i \in \mathbb{F}_2[x_1, x_2, \ldots, x_n], 1 \leqslant i \leqslant m$ be polynomials over the field $\mathbb{F}_2$ as input instance to the Poly-Eqs problem. The problem is denoted k-Poly-Eqs when the degrees are bounded by k which generalizes k-Sub-Sat as already explained in the introduction.

The unrestricted degree case is significantly different, because we can easily combine the m equations into a single equation as follows. Define

$$P = 1 + \prod_{i=1}^{m}(1 + P_i).$$

Clearly, the system $P_i = 0, 1 \leqslant i \leqslant m$ has a solution iff $P = 0$ has a solution.

Thus, assuming SETH, there is no algorithm essentially faster than 2^n for solving $P = 0$.

▶ Remark 24. There is also the question of how the polynomials P_i are given as part of the input. If $\deg P_i \leqslant k$ for all P_i then we can in polynomial-time compute their sparse representation as a linear combination of the n^k many monomials of degree at most k. However, in the above reduction of combining the P_i into a single polynomial, P is a small arithmetic formula. In fact, for the case of Poly-Eqs we consider, where the instance is a system of equations $P_i = 0, 1 \leqslant i \leqslant m$ such that $m = O(n)$ and each P_i has constant degree irreducible factors, we can assume that the P_i are given as arithmetic circuits.

We now show that Poly-Eqs instances $P_i = 0, 1 \leqslant i \leqslant m$ can be solved faster than 2^n if m is linear in n and the irreducible factors of each P_i are of constant degree. This can be seen as a "polynomial equations" analogue of Schuler's Sat algorithm for spare instances with unrestricted clause width [24, 5]. We note that a different degree reduction method, based on a rank argument, is used in [18, Section 4] to solve systems of polynomial equations $p_i = 0$, where each p_i is given by a sum of product of affine linear forms.

▶ **Theorem 25.** *Let $P_i = 0, 1 \leqslant i \leqslant c \cdot n$, for a constant $c > 0$, be an instance of* Poly-Eqs*, such that the degree of each irreducible factor of each P_i is bounded by a constant b. There is a randomized algorithm for* Poly-Eqs *that runs in time $2^{n(1-\alpha)}$ for such instances, where $\alpha > 0$ is a constant that depends on c and b.*

References

1 Vikraman Arvind and Venkatesan Guruswami. Cnf satisfiability in a subspace and related problems, 2021. `arXiv:2108.05914`.

2 Andreas Björklund, Petteri Kaski, and Ryan Williams. Solving systems of polynomial equations over GF(2) by a parity-counting self-reduction. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

3 Joshua Brakensiek, Sivakanth Gopi, and Venkatesan Guruswami. CSPs with global modular constraints: algorithms and hardness via polynomial representations. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 590–601, 2019.

4 Joshua Brakensiek and Venkatesan Guruswami. Bridging between 0/1 and linear programming via random walks. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, pages 568–577, 2019.

5 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 252–260. IEEE Computer Society, 2006.

6 Timothy M. Chan and R. Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. *ACM Trans. Algorithms*, 17(1):2:1–2:14, 2021.

7 Ruiwen Chen and Rahul Santhanam. Satisfiability on mixed instances. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 393–402, 2016.

8 Miguel Couceiro, Lucien Haddad, and Victor Lagerkvist. Fine-grained complexity of constraint satisfaction problems through partial polymorphisms: A survey. In *2019 IEEE 49th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 170–175, 2019.

9 Itai Dinur. Improved algorithms for solving polynomial systems over GF(2) by multiple parity-counting. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2550–2564. SIAM, 2021.

10 Rodney G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer Publishing Company, Incorporated, 2012.

11 J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series).* Springer-Verlag, 2006.

12 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972. SIAM, 2012.

13 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

14 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

15 Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. Strong partial clones and the time complexity of SAT problems. *J. Comput. Syst. Sci.*, 84:52–78, 2017.

16 Peter Jonsson, Victor Lagerkvist, and Biman Roy. Fine-grained time complexity of constraint satisfaction problems. *ACM Trans. Comput. Theory*, 13(1):2:1–2:32, 2021.

17 Victor Lagerkvist and Magnus Wahlström. Which NP-hard SAT and CSP problems admit exponentially improved algorithms? *CoRR*, abs/1801.09488, 2018. `arXiv:1801.09488`.

18 Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2190–2202, 2017.

19 Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discret. Appl. Math.*, 10(3):287–295, 1985.

20 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075, 2010.

21 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 566–574, 1997.

22 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978.

23 Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science*, pages 410–414, 1999.

24 Rainer Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms*, 54(1):40–44, 2005.

25 Mate Soos, Stephan Gocht, and Kuldeep S. Meel. Tinted, detached, and lazy CNF-XOR solving and its applications to counting and sampling. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, pages 463–484. Springer, 2020.

26 Mate Soos and Kuldeep S. Meel. BIRD: engineering an efficient CNF-XOR SAT solver and its applications to approximate model counting. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 1592–1599. AAAI Press, 2019.

27 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.