Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 159 (2013)

$(2+\varepsilon)$-SAT is NP-hard*

Per Austrin† Venkatesan Guruswami‡ Johan Håstad§

Abstract

We prove the following hardness result for a natural promise variant of the classical CNF-satisfiability problem: Given a CNF-formula where each clause has width w and the guarantee that there exists an assignment satisfying at least $g = \lceil \frac{w}{2} \rceil - 1$ literals in each clause, it is NP-hard to find a satisfying assignment to the formula (that sets at least one literal to true in each clause). On the other hand, when $g = \lceil \frac{w}{2} \rceil$, it is easy to find a satisfying assignment via simple generalizations of the algorithms for 2-SAT.

Viewing 2-SAT $\in$ P as easiness of SAT when 1-in-2 literals are true in every clause, and NP-hardness of 3-SAT as intractability of SAT when 1-in-3 literals are true, our result shows, for any fixed $\varepsilon > 0$, the hardness of finding a satisfying assignment to instances of "$(2+\varepsilon)$-SAT" where the density of satisfied literals in each clause is promised to exceed $\frac{1}{2+\varepsilon}$.

We also strengthen the results to prove that given a $(2k+1)$-uniform hypergraph that can be 2-colored such that each edge has perfect balance (at most $k+1$ vertices of either color), it is NP-hard to find a 2-coloring that avoids a monochromatic edge. In other words, a set system with discrepancy 1 is hard to distinguish from a set system with worst possible discrepancy.

Finally, we prove a general result showing intractability of promise CSPs based on the paucity of certain "weak polymorphisms." The core of the above hardness results is the claim that the only weak polymorphisms in these particular cases are juntas depending on few variables.

Key Words and Phrases: constraint satisfaction, complexity dichotomy, discrepancy, hypergraph coloring, polymorphisms, probabilistically checkable proofs.

*A preliminary version of this paper has appeared at FOCS 2014 [AGH14].

†School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden. Supported by ERC Advanced Grant 226203 and Swedish Research Council Grant 621-2012-4546. austrin@kth.se

‡Computer Science Department, Carnegie Mellon University, Pittsburgh, USA. Research supported in part by a Packard Fellowship, and NSF grants CCF-1115525 and CCF-1526092. guruswami@cmu.edu

§School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden. Supported by ERC Advanced Grant 226203, and some of this work done while visiting the Simon's institute. johanh@kth.se

ISSN 1433-8092

1 Introduction

One of the first distinctions we learn in complexity theory is that while 2-SAT can be solved in polynomial time, 3-SAT is our favorite NP-complete problem. As there are no integers between 2 and 3 this seems to be a sharp characterization but a closer inspection shows that a more fine-grained analysis is possible. One conclusion of the current paper is that the transition from easy to hard takes place *just after two* and not just before three.

Suppose we consider w-CNF formulas where each clause is of width exactly w and ask for an assignment that satisfies a literals in each clause. It follows more or less immediately from the above facts that this problem is NP-hard for $a \leqslant w-2$ and solvable in polynomial time when $a \geqslant w-1$. Suppose, however, that we make this into a promise problem and guarantee that there is an assignment that satisfies g literals in each clause for some $g > a$. We call the resulting problem (a,g,w)-SAT. It turns out that (a,g,w)-SAT can be solved in polynomial time if and only if $(a+1,g+1,w+1)$-SAT can be solved in polynomial time (we give the short proof of this fact in the preliminaries). We can thus focus on the case $a = 1$, i.e., finding satisfying assignments in the usual sense.

Let us start with some easy observations. Starting with a 3-CNF formula we can turn this into a $3g$-CNF formula by taking the union of all g-tuples of clauses. It is easy to see that if the original formula is satisfiable then we can satisfy g literals in each clause of the produced formula. From this it follows that $(1,g,w)$-SAT is NP-hard whenever $g \leqslant w/3$. To the best of our knowledge, no hardness was known for any $g > w/3$.

On the algorithmic side it is not difficult to see that the probabilistic 2-SAT algorithm of Papadimitriou [Pap91] extends to $(1,g,w)$-SAT when $g \geqslant w/2$. Using a linear program we show how to construct an algorithm running in deterministic polynomial time for the same range.

1.1 Our results

It turns out that this is all that can be achieved in polynomial time. The main new result of this paper is to establish that the problem is NP-hard whenever $g < w/2$. In particular, we have the following theorem. While the above discussion focused on the search problem, our hardness result applies even for the promise decision problem, of distinguishing instances of SAT that admit assignments satisfying many literals in each clause from unsatisfiable instances.

Theorem 1.1. *For every fixed integer $g \geqslant 1$, $(1,g,2g+1)$-SAT is NP-hard.*

This hardness result is the source of the above claim that the transition from easy to hard takes place at two. Once the density of satisfied literals drops strictly below one half, it is hard to find a satisfying assignment.[1]

To establish this result we give a reduction from the Label Cover problem, the usual starting point for inapproximability results (even though our results apply only to traditional constraint satisfaction problems where we want to satisfy all constraints). At first glance it might

[1]This is the sense in which we implied $(2+\varepsilon)$-SAT is NP-hard in the paper title, but we should mention here that $(2+\varepsilon)$-SAT has been used previously to denote instances of satisfiability containing a mix of 2CNF and 3CNF clauses, with about ε fraction of clauses being 3CNF [AKKK01]. As the terminology $(2+\varepsilon)$-SAT is restricted to just the title of this paper, we hope it does not cause much confusion.

seem slightly surprising that such a strong starting point is needed for our NP-hardness result. As an indication that something non-trivial is going on we give a proof that there is no standard gadget reduction from 3-SAT to any of our problems, and in particular not to $(1, g, 2g+1)$-SAT. This impossibility result is due to Dominik Scheder, and extends to show that there is no gadget reduction from $(1, g, 2g+1)$-SAT to $(1, g', 2g'+1)$-SAT for $g' > g \geqslant 1$.

One can also consider an approximation problem associated with $(1, g, 2g+1)$-SAT, where we are guaranteed that there is an assignment that satisfies at least g literals in a fraction c of clauses and the goal is to find an assignment that satisfies a fraction s of the clauses. Our proof of Theorem 1.1 implicitly shows that this problem is hard for $c = 1$ and some $s = s(g) < 1$, thus showing APX-hardness of $(1, g, 2g+1)$-SAT. We observe that an application of the theorem on "uselessness of predicates" from [AH13] implies a strong inapproximability result for this problem (albeit only for almost satisfiable instances and assuming the Unique Games conjecture) that shows hardness for $c = 1 - \varepsilon$ and $s = 1 - 2^{-(2g+1)} + \varepsilon$ for any $\varepsilon > 0$ (see Theorem 3.8).

Hypergraph discrepancy. A problem closely related to (a, g, w)-SAT is *hypergraph discrepancy*. Here, given sets of size $2g+1$ of elements from a universe, the problem is to color the elements with two colors such that each set has a good balance of colors. We extend our methods to prove the following hardness result, showing that systems of bounded-size sets with smallest possible discrepancy are NP-hard to distinguish from set systems with worst possible discrepancy.

Theorem 1.2. *For each fixed $g \geqslant 1$, given a $(2g+1)$-uniform hypergraph that admits a 2-coloring under which each hyperedge is evenly balanced (g elements of one color and $g+1$ of the other), it is NP-hard to find a 2-coloring that avoids creating a monochromatic hyperedge.*

The above result implies Theorem 1.1 via a simple reduction: for each hyperedge $(x_1, x_2, \ldots, x_{2g+1})$ of the hypergraph, create two width $2g+1$ clauses $(x_1 \vee x_2 \vee \cdots \vee x_{2g+1})$ and $(\overline{x_1} \vee \overline{x_2} \vee \cdots \vee \overline{x_{2g+1}})$. But we prove Theorem 1.1 first to illustrate our approach in a simpler setting that allows arbitrary clauses of width $2g+1$.

For the case of even-sized sets, i.e., $2g$-uniform hypergraphs for $g \geqslant 2$, we can get the following statement by a simple reduction from Theorem 1.2 (the first statement follows by adding a special private element to each set in the instance, and the second by taking all $2g$-element subsets of each $(2g+1)$-sized set in the instance).

Corollary 1.3. *(i) For each fixed $g \geqslant 2$, given a $2g$-uniform hypergraph that admits a 2-coloring under which each hyperedge is perfectly balanced (has g elements of each color), it is NP-hard to find a 2-coloring with discrepancy smaller than $(2g-2)$ (i.e., with more than one vertex of each color in every hyperedge).*

(ii) For each fixed $g \geqslant 2$, given a $2g$-uniform hypergraph that admits a 2-coloring with discrepancy at most 2, it is NP-hard to find a 2-coloring that avoids creating a monochromatic hyperedge.

The above statements are best possible in the sense that if there is a perfectly balanced (discrepancy 0) coloring of a $2g$-uniform hypergraph, then one can efficiently find a 2-coloring that avoids monochromatic hyperedges. This follows from the more general statement that $(1, g, 2g)$-SAT can be solved in polynomial time (via the reduction mentioned after Theorem 1.2).

For systems with unbounded set size, it is known that even if there is a coloring with discrepancy 0, it is NP-hard to find a coloring of discrepancy at most $c\sqrt{N}$, where $c > 0$ is a fixed constant and N is the size of the universe [CNN11].

Applications. Apart from the inherent interest in the given problems it is our hope that these new results will be useful as starting points for reductions to give new inapproximability results. While the inapproximability of Max-3Lin [Hås01] is a good starting point for problems where we are counting the number of satisfied constraints, our new problems might be good starting points when it is the *worst local situation* that governs the quality of a solution. As an example, consider the discrepancy problem, where the objective function is governed by the hyperedge with the most unbalanced coloring. As a small step in this direction we use our result to improve the inapproximability result for hereditary discrepancy for matrices from 3/2 to any number arbitrarily close to 2. Our Theorem 1.2 was also used recently in [FMS+14] to show a tight factor 2 inapproximability for a certain scheduling problem on two machines.

1.2 Techniques

Our hardness results are established via reductions from the Label Cover problem, which is an arity two constraint satisfaction problem over a large (but constant-sized) domain $[L] = \{1, 2, \ldots, L\}$ and the constraint relations allowed are functions $[L] \to [L]$. Specifically, an instance of Label Cover can be viewed as a (bipartite) graph $H = (U, V, E)$, and for each edge $e = (u, v) \in E$ there is a projection $\pi_e : [L] \to [L]$ stipulating that the value assigned to u maps to the value assigned to v under π_e. That is, an assignment $\sigma : U \cup V \to [L]$ satisfies the constraint given by edge e if $\pi_e(\sigma(u)) = \sigma(v)$. Given the current technology of reductions combining Label Cover and dictatorship tests, and the goal of only establishing hardness of deciding satisfiability (rather than some tight inapproximability ratio), the technical details of our proof are not difficult. Indeed, our result can be viewed as a particularly good entry point to understand the highly influential "Label cover + Long Code" framework for showing hardness results.

The combinatorial core of our results is a dictatorship test, where we are given a function $f : \{0,1\}^L \to \{0,1\}$, which we assume to be odd (i.e., $f(z) \neq f(\neg z)$ where $\neg z$ is the point antipodal to z) and the goal is to ascertain if it is a dictator, i.e., $f(z) = z_i \; \forall z \in \{0,1\}^L$, for some coordinate $i \in [L]$. For the problem of $(1, g, 2g+1)$-SAT, the allowed tests involve querying carefully chosen subsets of $2g+1$ points of the hypercube, $x_1, x_2, \ldots, x_{2g+1}$, and checking that at least g of the values $f(x_i)$ are 1. Given our desire to have all dictator functions pass all tests, there is a very natural choice for which subsets to pick, namely all $(2g+1)$-tuples for which in every coordinate there are at least g 1's. We prove that if the clauses produced by this natural dictatorship test are all satisfied by some f, then the function f is a junta that only depends on a constant number of inputs (independent of L).

To use this in a reduction from Label Cover, we have a hypercube of variables associated with each vertex $w \in U \cup V$ of the Label Cover instance. A Boolean assignment to these variables is naturally viewed as a function $f_w : \{0,1\}^L \to \{0,1\}$. In the completeness case, if the vertex w is assigned a label $\ell \in [L]$, then one takes f_w to be the dictator function (aka long code) that outputs the ℓ'th bit of the input, and the reduction ensures that this assignment will satisfy at least g literals in all the clauses of the produced $(2g+1)$-SAT instance.

For the soundness, the reduction will impose the above dictatorship test on each f_w, and

this will ensure that each f_w must be a junta in order to satisfy all these "intra-hypercube" clauses. Thus each f_w can be viewed as highlighting a small set of possible labels to w. To capture the projection constraints π_e, we also introduce some natural clauses between the hypercubes corresponding the endpoints u, v of each edge e. To pass the constraints that go across two long codes f_u, f_v we show that there must be some consistency between the set of labels highlighted by these juntas, and hence they can be used as labels to satisfy a constant fraction of the constraints in the Label Cover instance. Thus if the original Label Cover instance was highly unsatisfiable, there cannot be a satisfying assignment for the produced $(2g+1)$-SAT instance produced by the reduction.

Note that our proof method departs from the usual analytic approaches to analyze dictatorship tests (which seem ill-suited in our context) and relies on more combinatorial reasoning. The fact that our proofs are short and self-contained and yet yield a non-trivial hardness result about such a natural and easy to state variant of SAT is, in our opinion, one of the main selling points of this paper.

Connection to weak polymorphisms. *Polymorphisms* are operations that preserve a set of relations. They are a crucial tool in the algebraic approach to classifying the complexity of constraint satisfaction problems as either tractable or NP-hard. Formally, a polymorphism for a predicate $P \subseteq \{0,1\}^k$ is a map $f : \{0,1\}^L \to \{0,1\}$, for some arity L, such that for all choices of $x^{(1)}, x^{(2)}, \ldots, x^{(L)} \in P$, applying f componentwise to the $x^{(i)}$'s yields a string that also satisfies P; that is, if $z \in \{0,1\}^k$ is defined by $z_j = f(x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(L)})$ for $j = 1, 2, \ldots, k$, then $z \in P$ as well.

Our analysis of the dictatorship test (for $(1, g, 2g+1)$-SAT) can be stated equivalently as characterizing the "weak polymorphisms" that map assignments satisfying at least g out of $2g+1$ variables to an assignment satisfying at least one of those variables. In other words, the constraint imposed on the output of the polymorphism is a more relaxed one. More precisely, a weak polymorphism f for $(1, g, 2g+1)$-SAT of arity L is a map with the following property: if $x^{(1)}, x^{(2)}, \ldots, x^{(L)} \in \{0,1\}^{2g+1}$ are such that each $x^{(j)}$ has Hamming weight at least g, then applying f componentwise to the $x^{(j)}$'s yields a string z with at least one 1. By our combinatorial result, such a weak polymorphism must be a *junta*, that depends on at most $2g-1$ variables.

To elucidate the more general underlying principle governing our hardness results, we show that if the only weak polymorphisms mapping satisfying assignments of a predicate P to those of an implied predicate Q are juntas (i.e., depend on a fixed constant number of variables), than the promise CSP associated with the pair of predicates P, Q (where we are promised satisfiability according to P and the goal is to find an assignment satisfying the more lax predicate Q) is NP-hard.

1.3 Subsequent work

Our work raises several interesting directions for further research, some of which have already seen progress.

One way to generalize Theorem 1.2 would be to show hardness of weakly 2-coloring $(k+1)$-uniform hypergraphs that have even richer structure than discrepancy 1, for instance when the hypergraph is *k-rainbow colorable*, i.e., there is a k-coloring of vertices such that ev-

ery hyperedge has vertices of *each* color.[2]. If the hypergraph is k-uniform (instead of $k+1$-uniform), one can efficiently 2-color it without monochromatic hyperedges (e.g. via a random walk algorithm similar to the one in Section 6). Proving such a hardness for $k+1$-uniform hypergraphs is still open. However, in [GL15], the following strong inapproximability result for coloring hypergraphs has been shown (taking $g = 2$ below shows hardness of 2-coloring k-rainbow colorable hypergraphs when the hyperedges have size $2k$):

> For arbitrary integers $g, k, C \geqslant 2$, given as input a gk-uniform hypergraph that is promised to have a k-coloring where each color appears at least $(g-1)$ times in every hyperedge, it is NP-hard to weakly C-color the hypergraph.

While this result does not imply Theorem 1.2, taking $k = 2$ it implies part (ii) of Corollary 1.3, and in fact a stronger form when an arbitrary constant number of colors are allowed in the soundness case (part (i) also follows by a simple reduction). The proof of the above result in [GL15] is significantly more involved than our proofs, relying on analytic machinery such as invariance principles [Mos10, Wen13] and reverse hypercontractivity. The techniques can also be adapted to yield a proof of Theorem 1.1. Establishing our main result on discrepancy (Theorem 1.2), however, seems currently out of reach of these analytic methods.

The framework of showing hardness by characterizing the structure of the concerned weak polymorphisms was applied in [BG16a] to graph coloring. In particular, it was shown there that it is NP-hard to tell if an input graph is k-colorable, or has chromatic number at least $(2k-1)$, for all $k \geqslant 3$. The combinatorial crux was a structural theorem about weak polymorphisms (for k vs. $2k-2$ coloring) showing them to be dictators corrupted by noise in a controlled way. A similar approach yielded results for 2-coloring hypergraphs with low *strong* chromatic number.

The authors of [BG16a] also undertook a more systematic investigation into the complexity of Boolean promise CSPs in [BG16b]. A broad goal here would be to classify, for pairs of predicate P, Q where P implies Q, whether the problem of distinguishing CSP instance satisfiable according to P from those that are unsatisfiable even when constraints are relaxed to Q, is tractable or NP-hard. The case when the involved predicates are all *symmetric* (i.e., membership in the predicate only depends on the Hamming weight of the string) is settled in [BG16b] (even when the promise CSP allows several pairs of predicates (P_i, Q_i)). Weak polymorphisms are the crucial concept in this dichotomy result; in fact, this paper confirms that weak polymorphisms precisely capture the complexity of a promise CSP by extending the "Galois correspondence" known for usual CSPs (see, for instance, the excellent survey [Che09]) to the world of promise CSPs. A simple and generic method to handle repeated literals is also presented in [BG16b]; we exploit this to simplify our original proofs, particularly that of Theorem 1.2 where repetitions easily allow one to encode negations (which are not available in the discrepancy setting).

1.4 Organization

An outline of the paper is as follows. We start with some definitions and preliminaries in Section 2. As our main contribution is on the hardness side, we discuss the hardness results

[2]Note that such a hypergraph surely has a balanced 2-coloring, by merging $\lfloor \frac{k}{2} \rfloor$ colors into one group and the remaining $\lceil \frac{k}{2} \rceil$ colors into another group, so such a result will indeed strengthen Theorem 1.2

first and defer the algorithmic results to Section 6 towards the end of the paper.

We start by presenting our hardness result for $(1, g, 2g+1)$-SAT in Section 3. In Section 4 we then take a broad perspective and introduce the notion of "weak polymorphisms" and show that the hardness result for $(1, g, 2g+1)$-SAT is a special case of a more general theorem saying that under certain conditions, a CSP that does not have weak polymorphisms of arbitrary large arities must be NP-hard. We then discuss our results for discrepancy problems (Theorem 1.2) and improved inapproximability for hereditary discrepancy in Section 5. Finally we end with some concluding remarks in Section 7.

2 Preliminaries

We start with some basic definitions.

Definition 2.1. A w-SAT formula is a CNF formula where each clause has *width* exactly w.

One detail to consider is whether we allow repeated literals or two literals corresponding to the same variable in a clause. It turns out that this distinction does not change the complexity of the problems we consider. Specifically, in the follow-up work by Brakensiek and Guruswami [BG16b], Appendix C, it is shown that the complexity of a promise-CSP of the kind studied here is not affected by allowing or disallowing repeated variables in constraints. Hence throughout the paper we will freely allow repeated variables, since this tends to simplify the presentation.

Definition 2.2. A w-SAT formula Φ is *strongly g-satisfiable* if there is an assignment to the variables such that at least g literals are true in every clause of Φ.

Definition 2.3. For $1 \leqslant a \leqslant g < w$, the (a, g, w)-SAT promise problem is as follows. The input is a w-SAT formula Φ and the goal is to accept instances Φ that are strongly g-satisfiable and reject instances that do not admit any assignment that strongly a-satisfies Φ.

We have defined the decision version above, and in the search version we are given a w-SAT formula Φ that is *guaranteed* to be strongly g-satisfiable and the goal is to find an assignment that strongly a-satisfies Φ.

Note that $(1, 1, w)$-SAT is the usual w-SAT problem. Let us start with a couple of simple observations.

Observation 2.4. *There is a polynomial time reduction from (a, g, w)-SAT to $(a, g, w+1)$-SAT*

Proof. For each old clause, create two new clauses extending it by a variable and its complement. □

Proposition 2.5. *For $a \geqslant 1$, the problems (a, g, w)-SAT and $(a+1, g+1, w+1)$-SAT are interreducible to each other in polynomial time and in particular one of them is polynomial-time solvable iff the other one is.*

Proof. We establish two easy reductions and start with the obvious one.

By adding a shared dummy variable to all clauses of an (a, g, w)-SAT instance it follows that (a, g, w)-SAT reduces to $(a+1, g+1, w+1)$-SAT.

For the reduction in the other direction, take all subclauses of size w of each clause of size $w+1$. It is readily verified that this gives a correct reduction. □

In view of the above proposition we can focus on the case $a = 1$, i.e., the problem of finding a satisfying assignment in a w-CNF formula when we are guaranteed that there is an assignment that satisfies at least g literals in each clause.

We now define the discrepancy problem underlying Theorem 1.2 formally. Let S be a subset of size $2g+1$ of some universe U. We say that $X \subseteq U$ splits S evenly if $|X \cap S| \in \{g, g+1\}$.

Definition 2.6. An instance of the g-DISCREPANCY problem consists of a collection of sets $S_1, \ldots, S_m \subset U$ each of size exactly $2g+1$, and the objective is to distinguish between

Yes: there is an $X \subseteq U$ that splits each S_i evenly.

No: for every $X \subseteq U$, some S_i is not split by X at all (i.e., $|X \cap S_i| \in \{0, 2g+1\}$).

Label Cover and Long Codes. Our reductions establishing hardness results fit in the standard form for Probabilistically Checkable Proofs (PCPs), commonly used to establish inapproximability results for maximum constraint satisfaction problems. In particular our reductions start from label cover and use the long code encoding of each label.

Definition 2.7. An instance $\Psi = (U, V, E, \{\pi_e : L_V \to L_U\})$ of Label Cover consists of a bipartite graph (U, V, E), label sets L_U and L_V for U and V, and for each edge $e \in E$ a map $\pi_e : L_V \to L_U$.

A labeling σ is a map that assigns for each $u \in U$ a label $\sigma(u) \in L_U$ and for each $v \in V$ a label $\sigma(v) \in L_V$. The labeling σ satisfies an edge $e = (u, v)$ if $\pi_e(\sigma(v)) = \sigma(u)$.

The value of a labeling σ is the fraction of edges satisfied by σ, and the value $\mathsf{Opt}(\Psi)$ of Ψ is the maximum value of any labeling.

Theorem 2.8 ([ALM+98, Raz98]). *For every $\varepsilon > 0$, there are L_U, L_V such that given a Label Cover instance Ψ with label sets L_U and L_V it is NP-hard to distinguish between $\mathsf{Opt}(\Psi) = 1$ and $\mathsf{Opt}(\Psi) \leqslant \varepsilon$.*

For an alphabet L and symbol $\ell \in L$, the *long code encoding* of ℓ is a function $f : \{0,1\}^L \to \{0,1\}$, represented by its truth table, where $f(x) = x_\ell$. Whenever negation is available in the CSP for which we try to prove a lower bound we assume that tables are odd and respect negation (in standard PCP terminology "are folded") i.e. that $f(\neg x) = \neg f(x)$ where $\neg$ is a negation operator that works both on bits and strings (by negating each bit individually). The oddness of f is ensured by storing, for each pair $(x, \neg x)$, only the value $f(x)$ and if $f(\neg x)$ is needed then $\neg f(x)$ is used instead. We note that this is possible for our results for satisfiability but not for the hypergraph discrepancy problem which does not allow negations in the constraints.

We use some standard notation in the paper. We let $e_i \in \{0,1\}^n$ be the unit vector with a one in position i and use $\oplus$ to denote exclusive-or. Thus if $x \in \{0,1\}^n$ is any assignment, x and $x \oplus e_i$ differ in exactly coordinate i.

3 NP-hardness of $(1, g, 2g+1)$-SAT

We now return to the goal of establishing that $(1, g, 2g+1)$-SAT is NP-hard. In what follows we write $w = 2g + 1$.

3.1 A Dictatorship Gadget

First, we construct a dictatorship gadget, which is an instance defined over 2^n variables, viewed as a function $f : \{0,1\}^n \to \{0,1\}$ which is, as discussed in the preliminaries, assumed to be folded.

The constraints on f are all clauses of the form $(f(x_1) \vee f(x_2) \vee \ldots \vee f(x_w))$ where $x_1, \ldots, x_w$ are such that for each $j \in [n]$, $\sum_{i=1}^{w} x_{i,j} \geqslant g$. In other words there are at least g ones in each coordinate.

The completeness of the gadget (stated below) follows by construction.

Lemma 3.1. *If f is a dictatorship function then it strongly g-satisfies the dictatorship gadget.*

The converse of the above lemma is only true in a weaker sense.

Lemma 3.2. *Any assignment f which is odd and satisfies the dictatorship gadget depends on at most $2g - 1$ variables.*

In fact, this lemma is sharp – as we shall note in Section 4.1, the majority of $2g - 1$ variables does satisfy the gadget. The essential part of the above lemma is contained in the following claim that we establish first.

Claim 3.3. *Suppose f is odd, depends on g different variables $i_1, \ldots, i_g$, and satisfies the dictatorship gadget. Then $f(z) = 1$ for all inputs z such that $z_{i_1} = \ldots = z_{i_g} = 1$.*

Proof. Suppose for contradiction that there is an input z such that $f(z) = 0$ yet $z_{i_j} = 1$ for all $j \in [g]$. Since f depends on variables $i_1, \ldots, i_g$, there are inputs $x_1, \ldots, x_g$ such that for each $j \in [g]$

$$f(x_j) = 1 \quad \text{and} \quad f(x_j \oplus e_{i_j}) = 0\,.$$

Now consider the clause

$$f(z) \vee f(\neg x_1) \vee f(x_1 \oplus e_{i_1}) \vee \ldots \vee f(\neg x_g) \vee f(x_g \oplus e_{i_g}). \tag{1}$$

Note that this clause might contain repeated literals but by [BG16b], Theorem C.1, allowing this does not change the complexity of the problem. Clearly, this clause is not satisfied by f, so if this clause appears in the gadget we have our desired contradiction. In other words, we have to show that in each coordinate $i \in [n]$ we have at least g ones.

For any coordinate $i \notin \{i_1, \ldots, i_g\}$ we have that coordinate i is 1 in exactly one of $\{\neg x_j, x_j \oplus e_{i_j}\}$, for a total of at least g ones. For the coordinate i_j, we have that for all $j' \neq j$, at least one of $\{\neg x_{j'}, x_{j'} \oplus e_{i_{j'}}\}$ has a 1 in coordinate i_j. Furthermore z has a one in coordinate j, for a total of at least g ones. □

It is now easy to prove Lemma 3.2.

Proof of Lemma 3.2. Suppose f depends on $2g$ distinct variables $i_1, \ldots, i_g, j_1, \ldots, j_g$. Let z be an input such that $z_{i_1} = \ldots = z_{i_g} = 1$ and $z_{j_1} = \ldots = z_{j_g} = 0$. By Claim 3.3, $f(z) = 1$ and $f(\neg z) = 1$, contradicting that f is odd. □

3.2 Reduction from Label Cover

Let $\Psi = (U, V, E, \{\pi_e : L_V \to L_U\})$ be a Label Cover instance. To each vertex $u \in U$ we associate a function $f_u : \{0,1\}^{L_U} \to \{0,1\}$ intended to be a dictator of the label ℓ_u of u, and similarly $f_v : \{0,1\}^{L_V} \to \{0,1\}$ for $v \in V$.

We add the following constraints:

- For each $u \in U$ (resp. $v \in V$), the dictatorship gadget on f_u (resp. f_v).
- Fix an edge $e = (u, v)$. Let $x_1, \ldots, x_g \in \{0,1\}^{L_U}$ be g inputs on the U side and $y_1, \ldots, y_{g+1} \in \{0,1\}^{L_V}$ be $g+1$ inputs on the V side. If for each $l \in L_V$ it holds that

$$\sum_{j=1}^{g} x_{j,\pi_e(l)} + \sum_{j=1}^{g+1} y_{j,l} \geqslant g$$

  we add the constraint

$$f_u(x_1) \vee \ldots \vee f_u(x_g) \vee f_v(y_1) \vee \ldots f_v(y_{g+1})$$

We also use folding to make sure that each f_u (and f_v) is odd.

Call the resulting formula Φ. The completeness is standard and follows immediately from the construction and the completeness of the dictatorship gadget.

Lemma 3.4 (Completeness)**.** *If* $\mathsf{Opt}(\Psi) = 1$ *then* Φ *is strongly* g*-satisfiable.*

We turn to the more interesting case of soundness.

Lemma 3.5 (Soundness)**.** *If* Φ *is satisfiable then* $\mathsf{Opt}(\Psi) \geqslant 1/(2g-1)^2$*.*

Proof. Fix a satisfying assignment $\{f_u\}, \{f_v\}$ to Φ. By the soundness of the dictatorship gadget (Lemma 3.2) every f_u and f_v depends on at most $2g-1$ variables.

For each variable, let $S_u \subseteq L_U$ (resp. $S_v \subseteq L_V$) be the set of variables that f_u (resp. f_v) depends on and we have the following claim.

Claim 3.6. *For every edge* $e = (u, v)$ *it holds that* $\pi_e(S_v) \cap S_u \neq \emptyset$*.*

Proof. Suppose for contradiction that $S_u \cap \pi_e(S_v) = \emptyset$. Let $x_1, \ldots, x_g \in \{0,1\}^{L_U}$ be some set of g inputs such that $f_u(x_j) = 0$ and $x_{j,l'} = 1$ for all $l' \in L_U \setminus S_u$, and similarly let $y_1, \ldots, y_{g+1} \in \{0,1\}^{L_V}$ be $g+1$ inputs such that $f_v(y_j) = 0$ and $y_{j,l} = 1$ for all $l \in L_V \setminus S_v$.

Let us now check that

$$\sum_{j=1}^{g} x_{j,\pi_e(l)} + \sum_{j=1}^{g+1} y_{j,l} \geqslant g \quad \forall\, l \in L_V\,. \tag{2}$$

Note that this would imply that $f_u(x_1) \vee \ldots \vee f_u(x_g) \vee \ldots \vee f_v(y_1) \vee \ldots f_v(y_{g+1})$ is a clause in Φ and by construction it is not satisfied, a contradiction. For $l \notin S_v$, (2) holds because $y_{j,l} = 1$ for $j = 1, 2, \ldots, g+1$. For $l \in S_v$, we have $\pi_e(l) \notin S_u$, and therefore $x_{j,\pi_e(l)} = 1$ for $j = 1, 2, \ldots, g$, and (2) again holds. □

We now finish the proof of Lemma 3.5. We construct a random labeling by picking a random label from S_u (resp. S_v) for each variable $u \in U$ (resp. $v \in V$) of Ψ. For each edge $e = (u, v)$ it follows from the claim that the probability that e is satisfied by this labeling is $\frac{1}{|S_u| \cdot |S_v|} \geqslant 1/(2g-1)^2$ implying the bound on $\mathsf{Opt}(\Psi)$. □

To finish the proof of Theorem 1.1 we now simply make sure to start with a Label Cover instance with soundness at most $(2g-1)^{-2}$ and then invoke Theorem 2.8.

Remark 3.7. The above proof can also be used to show that in soundness case, there is no assignment that satisfies more than $1 - \gamma(g)$ fraction of clauses, for some $\gamma(g) > 0$. The required soundness $1/(2g-1)^2$ for the Label Cover instance can be achieved with $|L_U|, |L_V| \leqslant \text{poly}(g)$, and thus the tables f_u, f_v have sizes that only depend on g. Therefore, if $\gamma(g)$ is sufficiently small, then for any assignment to Ψ satisfying more than a $1 - \gamma(g)$ fraction of clauses, most of the tables f_u, f_v will have all clauses satisfied, and the constructed labeling then satisfies enough Label Cover constraints.

3.3 Inapproximability under the Unique Games conjecture

We now note the following "approximation resistance" phenomenon associated with *almost-satisfiable* instances of $(1, g, 2g+1)$-SAT.

Theorem 3.8. *Let $g \geqslant 1$ be an integer. Assuming the Unique Games conjecture [Kho02], the following promise problem is hard for every $\varepsilon > 0$. Given an instance of $(2g+1)$-SAT (with no repetitions of variables allowed within a clause), distinguish between the following two cases:*

1. *There is an assignment that strongly g-satisfies $(1-\varepsilon)$ of the clauses, and*
2. *There is no assignment that satisfies a fraction $(1 - 2^{-2g-1} + \varepsilon)$ of the clauses.*

Note that we have a gap between the two cases both in terms of the predicate being imposed on the clauses, and the fraction of clauses satisfiable according to the respective predicates. Also, since a random assignment satisfies an expected fraction $1 - 2^{-2g-1}$ of the clauses, the inapproximability factor is tight.

Proof of Theorem 3.8. We observe that there is a pairwise independent distribution μ on $\{0,1\}^{2g+1}$ which is supported only on strings with at least g ones. This distribution appears in the proof of Theorem 5.2 in [CHIS12], and as it is easy to describe, we recall it for completeness. Let $p = \frac{1}{2g+2}$. We sample a string according to μ as follows: With probability p, sample the all 1's string, and with probability $1-p$, sample a string uniformly from those with exactly g ones. A simple argument (see [CHIS12, Thm 5.2]) shows that every pair of bits are uniformly distributed under μ. The existence of μ together with Theorem 1.3 of [AH13] (the hardness part of which is based on [AM09]) shows the following: Given an instance of $(2g+1)$-SAT admitting an assignment that strongly g-satisfies a fraction $(1-\varepsilon)$ of the constraints, it is Unique Games-hard to find an assignment for which the distribution of $(2g+1)$-bit substrings appearing in

the scope of various constraints in the instance is ε-far from uniform. In particular, this means that it is Unique Games-hard to find an assignment satisfying a fraction $(1 - 2^{-2g-1} + \varepsilon)$ of the clauses, as a random assignment satisfies an expected fraction $(1 - 2^{-2g-1})$ of the clauses. □

4 General Framework

In this section, we show that the principle underlying our hardness result for $(1, g, 2g+1)$-SAT applies more generally. Namely, the absence of certain "weak polymorphisms" of arbitrarily large arities implies hardness for the associated promise CSP.

The hardness for $(1, g, 2g+1)$-SAT from the previous section turns out to be a particular case based on the result (established in Lemma 3.2) that there are no weak polymorphisms of arity larger than $2g-1$ that take assignments strongly g-satisfying a width $(2g+1)$-clause and output assignments that satisfy the clause.

We present our result for constraint satisfaction problems over an arbitrary finite domain allowing for folding. It is not clear whether or not this assumption is essential or whether it can be removed (as we shall do for the case of discrepancy in Section 5)

Definition 4.1 (CSP)**.** For integers $q, k \geqslant 2$, a constraint satisfaction problem over domain $[q] = \{0, 1, \ldots, q-1\}$ and arity k, denoted $\mathsf{CSP}(P)$, is specified by a predicate $P : [q]^k \to \{0, 1\}$.

An instance of this CSP is given by a set of variables V, and a collection of constraints, each specified by (τ, a) for $\tau = (v_1, \ldots, v_k) \in V^k$ and $a = (a_1, \ldots, a_k) \in [q]^k$ (which requires that the constraint P applied to the "literals" $v_1 + a_1, \ldots, v_k + a_k$ is met). Accordingly, the instance is said to be satisfiable if there is an assignment $\sigma : V \to [q]$ such that $P(\sigma(v_1) + a_1, \ldots, \sigma(v_k) + a_k) = 1$ for all constraints (τ, a) of the instance, where $\sigma(v_i) + a_i$ is to be understood mod q.

The following promise problem generalizes the $(1, g, 2g+1)$-SAT problem to arbitrary predicates.

Definition 4.2. For predicates $P, Q : [q]^k \to \{0, 1\}$ such that P implies Q (i.e., $\forall x,\ P(x) \leqslant Q(x)$), the (P, Q)-CSP problem is the following promise problem:

> Given an instance of $\mathsf{CSP}(P)$, distinguish between "Yes instances" which are satisfiable as a $\mathsf{CSP}(P)$ instance, and "No instances" which are unsatisfiable even as a $\mathsf{CSP}(Q)$ instance.

We now define the notion of weak polymorphisms that map satisfying assignments for one predicate into one for a more relaxed predicate.

Definition 4.3. Let $P, Q : [q]^k \to \{0, 1\}$ be predicates such that $\forall x,\ P(x) \leqslant Q(x)$. For a positive integer m, a function $f : [q]^m \to [q]$ is said to be a *folded* (P, Q)*-weak polymorphism* if the following properties hold:

1. (Polymorphism property) For all $b_1, b_2, \ldots, b_m \in P^{-1}(1)$, we have
$$(f(b_{1,1}, b_{2,1}, \ldots, b_{m,1}), f(b_{1,2}, b_{2,2}, \ldots, b_{m,2}), \cdots, f(b_{1,k}, b_{2,k}, \ldots, b_{m,k})) \in Q^{-1}(1)\ .$$
2. (Foldedness)[3] For every $x \in [q]^m$ and $a \in [q]$, $f(x + (a, a, \ldots, a)) = f(x) + a$ where the addition is mod q.

[3]This is the generalization to larger domains of the concept of oddness of Boolean functions.

In what follows, we define the *arity* of a function $f : [q]^m \to [q]$ to be the smallest integer t for which f is a t-junta (i.e., depends only on t input coordinates); formally, the smallest t for which there exists a subset $S \subseteq \{1, 2, \ldots, m\}$ with $|S| = t$ and a function $g : [q]^t \to [q]$ such that for every $x \in [q]^m$, $f(x) = g(x_{|S})$. Thus dictator functions have arity 1.

4.1 $(1, g, w)$-SAT through the polymorphic lens

Before proceeding with stating our general NP-hardness result, let us briefly revisit the $(1, g, 2g + 1)$-SAT problem using the framework of weak polymorphisms in order to shed additional light on the problem.

The $(1, g, 2g + 1)$-SAT problem is the same as the (P, Q)-CSP problem where $P : \{0, 1\}^{2g+1} \to \{0, 1\}$ accepts all inputs with at least g ones, and $Q : \{0, 1\}^{2g+1} \to \{0, 1\}$ accepts all inputs with at least 1 one. What non-trivial folded (P, Q)-weak polymorphisms does this pair of predicates admit?

A natural family of polymorphisms are majority operations on some m number of variables.

Proposition 4.4. *For the predicates P and Q from the $(1, g, 2g + 1)$-SAT problem, the majority on m variables for odd m is a folded (P, Q)-weak polymorphism if and only if $m \leqslant 2g - 1$.*

Proof. Consider m bit strings of length $2g + 1$ with weight at least g. There are in total $g \cdot m$ ones, meaning that in some coordinate, at least $\lceil \frac{g \cdot m}{2g+1} \rceil$ of the m strings has a one. Thus if $\lceil \frac{g \cdot m}{2g+1} \rceil \geqslant \lceil m/2 \rceil$ then the majority of one of the $2g + 1$ coordinates will be true, so the $(2g + 1)$-bit string of majorities satisfies Q and m-bit majority is a weak polymorphism. Conversely if $\lceil \frac{g \cdot m}{2g+1} \rceil < m/2$ then it is possible to make m inputs of length $2g + 1$ and weight g such that in each coordinate there are less than $m/2$ ones, and m-bit majority is then not a weak polymorphism.

Since m is odd, the condition $\lceil \frac{g \cdot m}{2g+1} \rceil \geqslant \lceil m/2 \rceil$ can be rewritten as $\frac{g \cdot m}{2g+1} > \frac{m-1}{2}$ or equivalently $m < 2g + 1$. Using again that m is odd, this is equivalent with $m \leqslant 2g - 1$. □

With this perspective, Lemma 3.2 is a strengthening of the "only if" direction of Proposition 4.4 – it says that not only are majorities of large arities not weak polymorphisms for this pair of predicates, in fact no operations that depend on more than $2g - 1$ variables can be weak polymorphisms. It is this lack of complicated polymorphisms that directly drives the NP-hardness result for $(1, g, 2g + 1)$-SAT. This is in contrast to $(1, g, 2g)$-SAT, where it is readily verified that majorities of all odd arities are weak polymorphisms.

However, while $(1, g, 2g + 1)$-SAT does not have arbitrarily large weak polymorphisms, it does have some non-trivial polymorphisms – majorities up to $2g - 1$ variables. As we note next, even the existence of these polymorphisms has algorithmic implications. In particular, they imply that there is no simple gadget reduction from 3-SAT to $(1, g, 2g + 1)$-SAT.

Consider the possibility of such a gadget reduction. For each clause, say $(x_1 \vee x_2 \vee x_3)$, this hypothetical reduction introduces a number of auxiliary variables a_i (which are particular to this clause) and forms a number of constraints in the form of clauses of width $2g + 1$. This reduction must satisfy:

1. Completeness: for each assignment to (x_1, x_2, x_3) that satisfies the original clause there is an assignment to the auxiliary variables such that at least g literals in each new clause are true, and

2. Soundness: if x_1, x_2, x_3 are all set to false, no assignment to the auxiliary variables satisfies all the new clauses.

If we do not allow repeated literals in a clause, then, as pointed out to us by John Wright, it easy to see that no such reduction can exist. Namely take the assignment to the auxiliary variables when x_1 is true and x_2 and x_3 are false and use the same assignment when all the three variables are false. As x_1 only appears once in each clause this change can only decrease the number of true literals in a clause by one and hence each clause remains satisfied. It turns out that allowing repeated literals does not help and we have the following proposition pointed out to us by Dominik Scheder.

Proposition 4.5. *There is no gadget reduction from* 3-SAT *to* $(1, g, 2g+1)$-SAT.

Proof. Consider the three cases when (x_1, x_2, x_3) takes the values $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$, respectively. Consider the good assignment to the auxiliary variables in each of these three cases, satisfying at least g literals in each $(2g+1)$-clause. Define a new assignment to the x_i's and the auxiliary variables as the majority of these three assignments; note that x_1, x_2, x_3 are assigned 0 under this majority assignment. We claim that this new assignment satisfies at least one literal in each created clause even when the x_i's are all false.

To see this, look at a single clause of width $2g+1$ and consider the values of these $2g+1$ literals under the three assignments. Since majority of 3 variables is a weak polymorphism, it follows that at least one of the $2g+1$ literals must be true after applying majority, and since each of the original variables x_1, x_2, x_3 is set to false in the majority assignment, the true literal must be one of the auxiliary variables. □

It is not difficult to see that the argument can be extended to prove that there is no gadget reduction from $(1, g, 2g+1)$-SAT to $(1, g', 2g'+1)$-SAT for $g' > g$, meaning that these in some sense form a strict hierarchy of problems. Instead of the assignments $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$, we now take take $2g+1$ assignments, each assignment having g true literals, and each literal being true in g of the assignments. A majority of these assignments does the trick as in the above argument.

4.2 General NP-hardness result

We now state our general theorem (Theorem 4.7 below) connecting (the lack of) polymorphisms to the hardness of (P, Q)-CSP. This generalizes a similar well-known statement for the case $P = Q$, namely that if the only polymorphisms for the constraint P are dictators, then the CSP(P) problem is NP-hard. The converse of this statement would imply the algebraic dichotomy conjecture of [BJK05] that precisely ties the tractability of a CSP to the existence of non-trivial polymorphisms. Establishing the converse of Theorem 4.7 for the promise version is only harder, but an interesting question would be to try to prove it for Boolean CSPs where the complexity dichotomy was shown long ago by Schaefer [Sch78] (see the survey [Che09] for a modern algebraic treatment of Schaefer's dichotomy theorem).

The statement of the theorem imposes the following technical condition on P.

Definition 4.6. A predicate $P : [q]^k \to \{0,1\}$ is said to be *full-domain-using* if for every $i \in \{1,2,\ldots,k\}$ and $a \in [q]$, there is a satisfying assignment to P that sets the i'th variable to a.

Note that if P is not full-domain-using, say $P(x_1, x_2, \ldots, x_k) = 1$ implies $x_i \in T$, then the range of any variable that appears in the i'th position of any constraint can be reduced. In the case of Boolean variables this would determine the value of any such variable, but also in the general case simplifications can be made. Therefore, the full-domain-using property of P is a natural non-degeneracy condition to assume.

We are now ready to state the main result of this section, which we prove in the rest of the section.

Theorem 4.7 (Large arity polymorphisms are necessary for tractability)**.** *Suppose $P, Q : [q]^k \to \{0,1\}$ are predicates such that every folded (P,Q)-weak polymorphism has arity bounded by a finite constant B, and assume that P is full-domain-using. Then (P,Q)-CSP is NP-hard.*

4.3 Dictatorship test

As usual, we start with a dictatorship test for a function $f : [q]^m \to [q]$ with constraints corresponding to the (P,Q)-CSP problem. We assume that f is folded, i.e., $f(x + (a,a,\ldots,a)) = f(x) + a$ for every $x \in [q]^m$ and $a \in [q]$. The constraints of this test are as follows:

> For all $x_1, x_2, \ldots, x_k \in [q]^m$ such that $P(x_{1,j}, x_{2,j}, \ldots, x_{k,j}) = 1$ for each $j \in \{1,2,\ldots,m\}$, check that
> $$Q(f(x_1), f(x_2), \cdots, f(x_k)) = 1\,. \tag{3}$$

The completeness of the test is obvious by design.

Lemma 4.8. *If f is a dictatorship function, then it satisfies all the constraints* (3) *(even if the predicate Q is replaced with P in those constraints).*

It follows pretty much from definition that a function f which passes all the checks (3) is a (P,Q)-weak polymorphism. Indeed we can take m arbitrary satisfying assignments to P as the j'th entries of $x_1, x_2, \ldots, x_k$ for $j = 1,2,\ldots,m$, and the output $(f(x_1), \ldots, f(x_k))$ must satisfy Q. Therefore, we also have the soundness property, similar to Lemma 3.2:

Lemma 4.9. *If every folded (P,Q)-weak polymorphism has arity bounded by B, then any folded f that satisfies all constraints* (3) *of the dictatorship gadget depends on at most B variables.*

4.4 NP-hardness reduction

We now turn to using the above construction in a NP-hardness reduction. Instead of the "normal" bipartite Label Cover, we reduce from a k-partite version of Label Cover (where k is the arity of the predicates P, Q). This version was originally proposed and used by Feige [Fei98] for his tight inapproximability result for set cover.

Definition 4.10 (Multi-partite Label Cover)**.** An instance of k-partite Label Cover consists of a k-partite k-uniform hypergraph $(U_1, U_2, \ldots, U_k, E)$, label sets L and $\tilde{L}$, and constraint functions $\pi_i^{(e)} : L \to \tilde{L}$ for each hyperedge $e \in E$ and $1 \leqslant i \leqslant k$.

A labeling solution to such an instance consists of assignments $\sigma_i : U_i \to L$.

We say a hyperedge $e = (u_1, u_2, \ldots, u_k)$ is *strongly satisfied* by such a labeling if

$$\pi_1^{(e)}(\sigma_1(u_1)) = \pi_2^{(e)}(\sigma_2(u_2)) = \cdots = \pi_k^{(e)}(\sigma_k(u_k)) \,,$$

and *weakly satisfied* if for *some* pair (i, j), $1 \leqslant i < j \leqslant k$,

$$\pi_i^{(e)}(\sigma_i(u_i)) = \pi_j^{(e)}(\sigma_j(u_j)) \,.$$

The following inapproximability result for k-partite Label Cover was shown by Feige [Fei98].

Theorem 4.11. *Let $k \geqslant 2$ be an integer. For all $\varepsilon > 0$, there exists $\ell = \ell(k, \varepsilon)$ such that given a k-partite Label Cover instance with label sets of size at most ℓ, it is NP-hard to distinguish between the following two cases:*

1. *(Yes instance) There exists a labeling solution that strongly satisfies every edge.*
2. *(No instance) Every labeling solution weakly satisfies at most a fraction ε of the hyperedges.*

We now describe the reduction from k-partite Label Cover to (P, Q)-CSP. Suppose we are given an instance with hypergraph $(U_1, U_2, \ldots, U_k, E)$, label sets $L, \tilde{L}$, and constraint functions $\pi_i^{(e)}$. For each $u_i \in U_i$, we associate a function $f_{u_i} : [q]^L \to [q]$, which we assume to be folded, and which is intended to be a dictator of the label $\sigma_i(u_i)$ of u_i. We add the following constraints:

- For each $u_i \in U_i$, $i = 1, 2, \ldots, k$, the dictatorship gadget from Section 4.3 on f_{u_i}.
- For each hyperedge $e = (u_1, u_2, \ldots, u_k) \in E$ add the constraint

$$Q(f_{u_1}(x_1), f_{u_2}(x_2), \ldots, f_{u_k}(x_k)) = 1 \,, \tag{4}$$

  for every choice of $x_1, x_2, \ldots, x_k$ which satisfy

$$P(x_{1,l_1}, x_{2,l_2}, \ldots, x_{k,l_k}) = 1$$

  for all tuples $(l_1, l_2, \ldots, l_k) \in L^k$ with $\pi_1^{(e)}(l_1) = \pi_2^{(e)}(l_2) = \cdots \pi_k^{(e)}(l_k)$.

The completeness of the reduction follows immediately from the construction, by taking f_{u_i} to be the dictatorship functions corresponding to the label of u_i.

Lemma 4.12. *If there is a labeling to the k-partite Label Cover instance which strongly satisfies every hyperedge, then the above instance is satisfiable even as a $\mathsf{CSP}(P)$ instance (i.e., when replacing predicate Q with P in all the constraints).*

It remains to analyze the soundness of the reduction. This is established in Lemma 4.13 below. Note that by picking the soundness ε of the Label Cover instance to be $\ll 1/B^2$, Theorem 4.7 would follow from Theorem 4.11, and Lemmas 4.12 and 4.13.

Lemma 4.13. *Suppose every folded (P, Q)-weak polymorphism has arity at most B, and P is full-domain-using. Then, if the $\mathsf{CSP}(Q)$ instance produced by the above reduction is satisfiable, there is a labeling to the original k-partite Label Cover instance that weakly satisfies at least $1/B^2$ of the hyperedges.*

Proof. Suppose we have folded tables $f_{u_i} : [q]^L \to [q]$ for $u_i \in U_i$, $1 \leqslant i \leqslant k$, that satisfy all the constraints. Then by the soundness of the dictatorship tests, there must be subsets $S_{u_i} \subset L$ for each vertex u_i with $|S_{u_i}| \leqslant B$ such that f_{u_i} only depends on variables in S_{u_i}.

Fix an hyperedge $e = (u_1, u_2, \ldots, u_k) \in E$. For notational simplicity, denote S_{u_i} by S_i. We now prove that in order to satisfy all the constraints (4), we must have

$$\pi_i^{(e)}(S_i) \cap \pi_j^{(e)}(S_j) \neq \emptyset \text{ for some } i \neq j\,. \tag{5}$$

Once we prove this, the labeling strategy of assigning to each u_i a random label from S_{u_i} will weakly satisfy at least $1/B^2$ of the hyperedges in expectation, implying the existence of a labeling that weakly satisfies at least $1/B^2$ of the hyperedges of the k-partite Label Cover instance.

Suppose for contradiction that (5) is not the case and we have

$$\forall\, i, j;\ 1 \leqslant i < j \leqslant k;\ \pi_i^{(e)}(S_i) \cap \pi_j^{(e)}(S_j) = \emptyset\,. \tag{6}$$

Pick an assignment $\beta = (\beta_1, \beta_2, \ldots, \beta_k) \in Q^{-1}(0)$. (The absence of (P, Q)-weak polymorphisms of arbitrary arity implies that Q cannot be the trivial predicate always outputting 1.) For $i = 1, 2, \ldots, k$, pick $x_i \in [q]^L$ such that

1. x_i is constant on S_i, say $x_{i,l} = b_i$ for all $l \in S_i$, and
2. $f_{u_i}(x_i) = \beta_i$

Such a choice is possible as the f_{u_i} are folded. Note that by choice

$$Q(f_{u_1}(x_1), f_{u_2}(x_2), \ldots, f_{u_k}(x_k)) = 0\,. \tag{7}$$

Therefore, to get a contradiction we need to ensure that x_i's can be completed in other coordinates in a manner so that the test (4) is made with this choice of x_i's.

Define $T_i = (\pi_i^{(e)})^{-1}(\pi_i^{(e)}(S_i))$, i.e., the labels l' that collide with some $l \in S_i$ under $\pi_i^{(e)}$. Since $f_{u_i}(x)$ only depends on $x_{|S_i}$, we can further assume that the chosen x_i takes the constant value b_i for every coordinate in T_i. By (6), we have

$$\pi_i^{(e)}(T_i) \cap \pi_j^{(e)}(T_j) = \emptyset \text{ for all } 1 \leqslant i < j \leqslant k\,. \tag{8}$$

By the full-domain-using property of P, for $1 \leqslant i \leqslant k$, we can find assignments $\theta^{(i)} \in P^{-1}(1)$ such that $\theta^{(i)}$ takes value b_i in the i'th coordinate. We can now fill in the coordinates outside T_i in x_i, for each $i = 1, 2, \ldots, k$, as follows.

- For coordinates l of x_i such that $\pi_i^{(e)}(l) \notin \bigcup_j \pi_j^{(e)}(T_j)$, we set $x_{i,l} = a_i$ where $(a_1, a_1, \ldots, a_k)$ is some fixed satisfying assignment of P.

- For the coordinates l of x_i such that $\pi_i^{(e)}(l) \in \pi_j^{(e)}(T_j)$ for some $j \neq i$ (note that such a j, if it exists, is unique by (8)), we set $x_{i,l}$ to be the i'th coordinate of $\theta^{(j)}$.

One can check by a quick inspection that this construction creates a tuple $(x_1, x_2, \ldots, x_k)$ obeying the conditions under which the constraint (4) is added, which is in contradiction with (7). □

5 Hardness for Discrepancy Problems

The main result of this section is the following theorem, which is of course just an alternate statement of Theorem 1.2.

Theorem 5.1. *g-DISCREPANCY is NP-hard for every constant $g > 1$.*

The reduction and proof follows along the same lines as the hardness proof for $(1, g, 2g+1)$-SAT in Section 3, though some minor modifications in the constructions are needed since "true" and "false" are now treated symmetrically.

There are two differences between $(1, g, 2g+1)$-SAT and g-DISCREPANCY. The first one is that we no longer have the concept of negated literals and so we can no longer assume that our long codes are folded. Using repeated elements in our sets, this problem can be solved very simply by adding a constraint with g copies of $f(x)$ and $g+1$ copies of $f(\neg x)$ for all x. Using the generic method to eliminate repeated literals presented in [BG16b], we can convert this to a collection of $(2g+1)$-sized sets with no repetitions.

The second difference is that in the YES case in g-discrepancy, we want every $(2g+1)$-bit string to have weight either g or $g+1$, whereas in $(1, g, 2g+1)$-SAT the inputs just have weight at least g. Thus we adjust the dictatorship from Section 3.1 as follows. Recall that previously we would add a clause of the form $(f(x_1) \vee f(x_2) \vee \ldots \vee f(x_w))$ whenever $x_1, \ldots, x_w$ are such that for each $j \in [n]$, $\sum_{i=1}^{w} x_{i,j} \geqslant g$. Now we instead add a (multi)set $\{f(x_1), \ldots, f(x_w)\}$ to our set family whenever $x_1, \ldots, x_w$ are such that for each $j \in [n]$, $\sum_{i=1}^{w} x_{i,j} \in \{g, g+1\}$.

Note that in the new dictatorship gadget, multisets with g copies of $f(x)$ and $g+1$ copies of $f(\neg x)$ are included. This implies that any function f which does not leave any of the sets in the gadget monochromatic must be odd. The analysis of the dictatorship gadget now follows very closely along the lines of the previous analysis in Section 3.1. In particular we have the following analogue of Claim 3.3.

Claim 5.2. *Suppose f depends on g different variables $i_1, \ldots, i_g$ and does not leave any set of the discrepancy dictatorship gadget monochromatic. Then there are constants $c_1, \ldots, c_g \in \{0, 1\}$ such that $f(z) = 1$ for all inputs z such that $(z_{i_1}, \ldots, z_{i_g}) = (c_1, \ldots, c_g)$.*

Proof. Since f depends on the g variables i_1 to i_g, there are inputs $x_1, \ldots, x_g$ such that for each $1 \leqslant j \leqslant g$

$$f(x_j) = 1 \qquad\qquad f(x_j \oplus e_{i_j}) = 0$$

Now define $c_j = x_{j,i_j}$, the i_j'th bit of x_j. We claim that these values satisfy the desired property. To see this, suppose for contradiction that there is a z such that $(z_{i_1}, \ldots, z_{i_g}) = (c_1, \ldots, c_g)$ but $f(z) = 0$.

Consider the multiset

$$\{f(z), f(\neg x_1), f(x_1 \oplus e_{i_1}), \ldots, f(\neg x_g), f(x_g \oplus e_{i_g})\}$$

Clearly, this multiset is left monochromatic by f. However, it is also included in the dictatorship gadget: for any coordinate not among $i_1, \ldots, i_g$ each pair $(\neg x_j, x_j \oplus e_{i_j})$ contributes 1 zero, and 1 one, for a total of at least g of each. For coordinate i_j for some $1 \leqslant j \leqslant g$, the pairs $(\neg x_{j'}, x_{j'} \oplus e_{i_{j'}})$ for $j' \neq j$ contribute $g-1$ zeros and $g-1$ ones, and then the pair $(z, \neg x_j)$ contribute 1 zero and 1 one. □

Using this claim we immediately obtain a direct analogue of Lemma 3.2

Lemma 5.3. *Any assignment f which does not leave any set in the dictatorship gadget monochromatic depends on at most $2g-1$ variables.*

Using this gadget, we can now obtain a NP-hardness reduction from Label Cover. Since the proof is identical to the proof in Section 3.2, we leave the details to the reader.

5.1 Application to hereditary discrepancy

Given a family of sets $\mathcal{F} = \{S_1, \ldots, S_m\}$ over some universe U, and a subuniverse $U' \subseteq U$, let $\mathcal{F}|_{U'} = \{S \cap U' \mid S \in \mathcal{F}\}$ denote the set family where each set in $\mathcal{F}$ is restricted to U'. For each $U' \subseteq U$, let $\mathrm{disc}(\mathcal{F}|_{U'})$ denote the discrepancy of the restricted set family. The *hereditary discrepancy* of $\mathcal{F}$ is defined as $\max_{U' \subseteq U} \mathrm{disc}(\mathcal{F}|_{U'})$, the worst discrepancy of any restriction of $\mathcal{F}$.

The problem of computing hereditary discrepancy was considered in [NTZ13] where it was proved to be hard to approximate within a factor $\frac{3}{2}$. It follows more or less immediately from Theorem 5.1 that we can improve this inapproximability factor to $\frac{2g+1}{g+1}$ for any integer g and thus arbitrarily close to 2. Both the reduction of [NTZ13] and here actually gives a family such that either the discrepancy of $\mathcal{F}$ is large (in our case $2g+1$) or the hereditary discrepancy is small (in our case $g+1$).

To see this let $\mathcal{F}$ be set system constructed to prove Theorem 5.1. In the yes case $\mathcal{F}$ has a near-balanced 2-coloring, and since each set in $\mathcal{F}$ has size $2g+1$, this 2-coloring shows that any restriction of $\mathcal{F}$ has discrepancy at most $g+1$ and hence this is an upper bound on the hereditary discrepancy. In the no case $\mathcal{F}$ itself has discrepancy $2g+1$ and thus also this large hereditary discrepancy.

6 Algorithms for $(1, g, 2g)$-Sat

We now present efficient algorithms to find a satisfying assignment when at least half the literals in each clause are promised to be true under some assignment.

6.1 A randomized algorithm

Let us first describe a simple randomized algorithm closely following Papadimitriou's algorithm [Pap91] for 2-Sat.

Algorithm 1: Randomized algorithm for $(1, g, w)$-SAT.

(1) $x \leftarrow$ arbitrary assignment
(2) **while** x is not satisfying
(3) Pick (arbitrarily) a falsified clause ϕ
(4) Flip the value of a randomly chosen literal of ϕ
(5) **return** x

The analysis of this algorithm is essentially equivalent with that of Papadimitriou.

Proposition 6.1. *If Φ is a strongly g-satisfiable width-w CNF formula and $w \leqslant 2g$, then Algorithm 1 finds a satisfying assignment in $O(tn^2)$ steps with probability at least $1 - 2^{-t}$.*

Proof. Let x^* be any g-satisfying assignment, and let x_i be the value of x in the i'th iteration of Algorithm 1 and ϕ_i be the clause chosen. Define the random variable $D_i = d(x_i, x^*)$ where $d(x, y)$ is the Hamming distance between x and y. Clearly, $D_{i+1} - D_i = \pm 1$. Furthermore, since x^* satisfies g literals of ϕ_i and it contains at most $2g$ literals we have

$$\Pr[D_{i+1} = D_i - 1] \geqslant 1/2$$

so that $\mathbb{E}[D_{i+1} - D_i | D_i] \leqslant 0$. In other words $D_1, D_2, \ldots$ describes a random walk starting at some point between 0 and n where each step is unbiased or biased towards 0. This is a "gambler's ruin" chain with reflecting barrier (because the distance cannot increase beyond n). In such a walk, the gambler is broke (i.e., the distance hits 0) in n^2 steps with constant probability. The probability that it fails to hit 0 with ctn^2 steps is thus at most 2^{-t} for a suitable chosen constant c. □

Note that this algorithm is not affected by the presence of multiple copies of the same literal within a clause. Also note that if $w < 2g$ the walk is in fact biased towards 0 and a satisfying assignment is, with high probability, found in $O(n)$ steps.

We next present a deterministic algorithm that is based on linear programming.

6.2 A deterministic algorithm

There is a very natural linear program connected to a w-Sat formula. Namely, relax each Boolean variable x_i to a real-valued variable y_i which takes values in $[0, 1]$. In the formula replace x_i by y_i and $\neg x_i$ by $1 - y_i$ and require that the sum over each clause is at least g. As an example in $(1, 2, 4)$-SAT we replace the clause $(x_1 \vee x_2 \vee \neg x_3 \vee \neg x_4)$ by the linear inequality

$$y_1 + y_2 + (1 - y_3) + (1 - y_4) \geqslant 2$$

This might not seem like a very useful linear program as $y_i = 1/2$ for all i satisfy all the inequalities when $w = 2g$, but forcing a single variable to take the value 0 or 1 does give useful information. Consider the procedure described in Algorithm 2 where we let $\lfloor y \rceil$ denote the integer closest to y (we only apply this operation to numbers whose fractional part is not $1/2$ and hence this number is unique). We establish in the below proposition that the algorithm is indeed correct.

Algorithm 2: Deterministic algorithm for $(1, g, w)$-SAT.

(1) **repeat**
(2) Let x_i be some unassigned variable
(3) Choose $b \in \{0, 1\}$ such that the basic LP with y_i forced to b is feasible
(4) **if** the LP is infeasible for both choices of b
(5) **return** "Not strongly g-satisfiable"
(6) Let $y_1, \ldots, y_n$ be the LP solution when y_i is forced to b
(7) **foreach** i such that $y_i \neq \frac{1}{2}$
(8) Assign $x_i \leftarrow \lfloor y_i \rceil$
(9) Remove all satisfied clauses from the formula
(10) **until** all variables are assigned
(11) **return** x

Proposition 6.2. *Given a strongly g-satisfiable w-SAT instance, where $w \leqslant 2g$, Algorithm 2 finds a satisfying assignment.*

Proof. Note first that if $w < 2g$ then in the LP solution any clause must contain a literal whose value is greater than $1/2$ and thus in fact the tentative assignment to x_i in line 3 is not needed.

When $w = 2g$ each clause that contains a literal that is not exactly $1/2$ must, in each feasible solution, contain a literal that is of value strictly greater than $1/2$. This implies that if we assign the value of some variable in a clause then in the same round we set one of its literals to true and satisfy the clause. Thus there is no risk of falsifying a clause during this process. In addition, the clauses that remain after each round consist only of unassigned variables and thus the remaining set of clauses still forms a strongly g-satisfiable instance. □

7 Conclusions

We have given a sharp classification for a natural promise version of CNF-Sat. As CNF-Sat is a favorite starting point for many reductions we hope that this can give quantitatively improved results in many situations. We gave a rather modest example in Section 5.1 but one might hope that there are many other possibilities.

As general framework from Section 4 showed, the *non-existence* of weak polymorphisms whose outputs satisfy a weaker predicate Q than the predicate P obeyed by its inputs implies the *hardness* of finding a Q-satisfying assignment to a P-satisfiable CSP instance. Can one establish results in the converse direction, obtaining *algorithms* based on the *existence* of non-trivial weak polymorphisms, at least for the case of Boolean predicates P, Q? (We recall that when $P = Q$, we have Schaefer's dichotomy theorem in the Boolean case [Sch78], and the existence of non-trivial polymorphisms precisely governs the tractability of the associated CSP.) In a follow-up work, a dichotomy theorem for promise Boolean CSPs was established in the special case when the predicates involved are *symmetric* [BG16b], and the tractable cases are governed by essentially three different classes of non-trivial weak polymorphisms.

One may also consider an approximate version of (a, g, w)-SAT where we are guaranteed that there is an assignment that strongly g-satisfies a fraction c of clauses and the goal

is to find an assignment that strongly a-satisfies a fraction s of clauses. As mentioned in Section 3.3, we have a strong hardness result under the Unique Games Conjecture, wherein given a $(2g+1)$-SAT instance admitting an assignment that strongly g-satisfies a fraction $1-\varepsilon$ of the constraints, it is hard to find an assignment that satisfies a fraction $1-2^{-(2g+1)}+\varepsilon$ of the constraints (which is what a random assignment would achieve). Obtaining such a result without relying on the Unique Games Conjecture seems out of reach with current techniques. It is also an interesting goal to obtain a strong inapproximability result for this problem with perfect completeness, i.e., when the instance admits a strongly g-satisfying assignment.

Acknowledgments

We are indebted to Dominik Scheder for the claim about the non-existence of gadget reductions described in Section 4.1 and to John Wright for pointing out that the argument can be made very simple if we have no repeated literals. We thank the anonymous reviewers and the handling editor for their careful reading of the paper and several useful comments on the presentation.

References

[AGH14] Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2+\varepsilon)$-Sat is NP-hard. In *Proceedings of 55th Annual IEEE Symposium of Foundations of Computer Science*, pages 1–10, 2014. 1

[AH13] Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Transactions on Computation Theory*, 5:1–24, 2013. 3, 11

[AKKK01] Dimitris Achlioptas, Lefteris M. Kirousis, Evangelos Kranakis, and Danny Krizanc. Rigorous results for random (2+p)-sat. *Theor. Comput. Sci.*, 265(1-2):109–129, 2001. 2

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and intractability of approximation problems. *Journal of the ACM*, 45:501–555, 1998. 8

[AM09] Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18:249–271, 2009. 11

[BG16a] Joshua Brakensiek and Venkatesan Guruswami. New hardness results for graph and hypergraph colorings. In *Proceedings of the 31st Conference on Computational Complexity (CCC)*, pages 14:1–14:27, 2016. 6

[BG16b] Joshua Brakensiek and Venkatesan Guruswami. Promise constraint satisfaction: Algebraic structure and a symmetric Boolean dichotomy. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:183, 2016. 6, 7, 9, 18, 21

[BJK05] Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. 14

[Che09] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1), 2009. 6, 14

[CHIS12] Mahdi Cheraghchi, Johan Håstad, Marcus Isaksson, and Ola Svensson. Approximating linear threshold predicates. *TOCT*, 4(1):2, 2012. 11

[CNN11] Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight hardness results for minimizing discrepancy. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1607–1614, 2011. 4

[Fei98] Uri Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998. 15, 16

[FMS+14] Esteban Feuerstein, Alberto Marchetti-Spaccamela, Frans Schalekamp, René Sitters, Suzanne van der Ster, Leen Stougie, and Anke van Zuylen. Scheduling over scenarios on two machines. In *Proceedings of the 20th International Conference on Computing and Combinatorics*, COCOON '14, pages 559–571, 2014. 4

[GL15] Venkatesan Guruswami and Euiwoong Lee. Strong inapproximability results on balanced rainbow-colorable hypergraphs. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 822–836, 2015. Full version to appear in Combinatorica. 6

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001. 4

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of 34th ACM Symposium on Theory of Computing*, pages 767–775, 2002. 11

[Mos10] Elchanan Mossel. Gaussian bounds for noise correlation of functions. *Geometric and Functional Analysis*, 19(6):1713–1756, 2010. 6

[NTZ13] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing*, STOC '13, pages 351–360, New York, NY, USA, 2013. ACM. 19

[Pap91] 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, 1991. 2, 19

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM J. on Computing*, 27:763–803, 1998. 8

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth annual ACM Symposium on Theory of Computing*, pages 216–226, 1978. 14, 21

[Wen13] Cenny Wenner. Circumventing d-to-1 for approximation resistance of satisfiable predicates strictly containing parity of width at least four. *Theory of Computing*, 9(23):703–757, 2013. 6

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il