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

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

Per Austrin* Venkatesan Guruswami† Johan Håstad‡

October 2013

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 can be viewed as showing, for every $\varepsilon > 0$, intractability of finding a satisfying assignment to instances of "$(2+\varepsilon)$-SAT" where the density of satisfied literals in each clause is $1/(2+\varepsilon)$.

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 in the wake of paucity of certain "weak polymorphisms." The core of the above hardness results is the claim that weak polymorphisms in these particular cases are juntas depending on few variables.

Keywords: Constraint satisfaction, complexity dichotomy, discrepancy, hypergraph coloring, polymorphisms, probabilistically checkable proofs.

*School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden. austrin@kth.se

†Computer Science Department, Carnegie Mellon University, Pittsburgh, USA. Research supported in part by NSF grant CCF-1115525. guruswami@cmu.edu

‡School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden, 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 $a = 1$.

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 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 [12] 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.

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.

Theorem 1.1. *For every 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.

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). It is slightly surprising that such a strong starting point is needed but 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 or 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 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 satisfies a fraction s fraction of clauses. We observe that an application of the theorem on "uselessness of predicates" from [2] 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 $\varepsilon > 0$.

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 $g \geqslant 1$, 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 edge.*

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 negated literals and repeated variables within a clause.

For the case of even-sized sets, i.e., $2g$-uniform hypergraphs for $g \geqslant 2$, a trivial gadget reduction from the above result ensures that it is hard to achieve discrepancy smaller than $(2g-2)$ even if there is a coloring with discrepancy 0. 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 $\Omega(\sqrt{N})$ where N is the size of the universe [5].

Connection to weak polymorphisms. The crux of our proof of Theorem 1.1 is to show that all *weak polymorphisms* which map assignments satisfying at least g out of $2g+1$ variables to an assignment satisfying at least one of those variables depend only on a small number of variables. (For Theorem 1.2, a slightly weaker statement that allows a small number of exceptional inputs is the basis for the proof.) To elucidate the underlying principle governing our hardness results, we show a general statement that a paucity of non-trivial weak polymorphisms leads to intractability of the associated promise CSP.

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 [9] 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 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.

Organization. An outline of the paper is as follows. We start with some definitions and preliminaries in Section 2. As the algorithmic results are relatively simple and our main contribution is on the hardness side, we present the algorithmic results in Section 7 towards the end of the paper, and discuss the hardness results first. Before presenting our hardness result for $(1, g, 2g+1)$-SAT in Section 4, we take a brief "polymorphic" detour in Section 3, proving the non-existence of a simple gadget reduction from 3-SAT to $(1, g, 2g+1)$-SAT, and also commenting on the general principle underlying our hardness result. This principle is governed by the lack of "weak polymorphisms" of large arities, and is described in detail in a self-contained Appendix A at the end of the paper. We present the extension of our hardness results to hypergraph discrepancy (Theorem 1.2) in Section 5, where we also show how to ensure that all variables appearing in a constraint are distinct. We give the application

to hereditary discrepancy in Section 6, and finally we end with some concluding remarks in Section 8.

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.

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. *The problems (a, g, w)-SAT and $(a+1, g+1, w+1)$-SAT are interreducible to each other in polynomial time. (Thus one of them is polynomial time solvable iff the other one is.)*

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

If we have an instance of (a, g, w)-SAT, then adding the same new variable to all clauses gives an instance of $(a + 1, g + 1, w + 1)$-SAT of the same difficulty.

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.

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 problem. While our algorithms apply directly to the general case, in the hardness results the requirement to use only distinct literals does create some slight technical complications. Hence we first give a cleaner proof of Theorem 1.1 where we allow repeated literals, and then present a full proof of the stronger result for hypergraph discrepancy, Theorem 1.2, in which we ensure that all variables in a constraint are distinct (i.e., the hypergraph is $(2g + 1)$-uniform). 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 ℓ is a map that assigns for each $u \in U$ a label $\ell(u) \in L_U$ and for each $v \in V$ a label $\ell(v) \in L_V$. The labeling ℓ satisfies an edge $e = (u, v)$ if $\pi_e(\ell(v)) = \ell(u)$.

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

Theorem 2.8 ([1, 13])**.** *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 $\mathsf{Opt}(\Psi) \leqslant \varepsilon$.*

The long code of $\ell \in L$ is a table $f : \{0,1\}^L \to \{0,1\}$ 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 A polymorphic detour

Before proceeding with the actual proof that $(1, g, 2g+1)$-SAT is NP-hard let us gather some intuition for the problem by proving that there is no simple gadget reduction from standard 3-SAT, and also comment on the polymorphic principle governing our hardness results (whose detailed description we defer to Appendix A).

Let us first consider the possibility of a gadget reduction from 3-SAT to $(1, 2, 5)$-SAT. 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 5. This reduction should satisfy:

1. Completeness: for each assignment to x_1, x_2 and x_3 that satisfies the original clause there is an assignment to the auxiliary variables such that at least two 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.

We have the following observation by Dominik Scheder that such a gadget reduction does not exist.

Proposition 3.1. *There is no gadget reduction from* 3-SAT *to* $(1, 2, 5)$-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 two literals in each created clause. Define a new assignment to the auxiliary variables as the majority of these three assignments. 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 5 and consider the values of these 5 literals under the three assignments. In total we have at least 6 satisfied literals. Thus, one of the five literals is satisfied in two of the three assignments, and this literal is then satisfied also in the majority assignment. □

It is not difficult to see that the proof extends to prove that there is no gadget reduction from $(1, g, 2g + 1)$-SAT to $(1, g', 2g' + 1)$-SAT for $g' > g$. We simply take $2g + 1$ assignments with g true literals in each such that each literal is true in g assignments. A majority of these assignments does the trick as in the above argument.

For readers familiar with the notion of polymorphisms from the CSP dichotomy theory, we note that what we have established is that majority of $2g + 1$ bits is a type of "weak polymorphism". Namely, if each input strongly g'-satisfies a clause of width $(2g' + 1)$ clause then the output must satisfy the clause if $g' > g$ but not for smaller values of g'. Moreover if $w \leqslant 2g$ then any odd majority is a polymorphism that takes assignments for width w clause that is strongly g-satisfying into a satisfying assignment. Our hardness result for $(1, g, 2g+1)$-SAT are based on a proof that such polymorphisms for $w > 2g$ can only depend on few variables.

One can consider the $(1, g, 2g+1)$-SAT problem as an instance of a large family of problems parametrized by two predicates P and Q of the same arity k, with P implying Q. One is given a large number of k-tuples of literals and the promise that there is an assignment such that all resulting k-tuples of bits satisfy P. The task now is to find an assignment such that the strings instead satisfy Q. It turns out that the lack of non-trivial polymorphisms on arbitrarily many variables that take satisfying assignments for P and map it to satisfying assignments for Q implies that this promise P vs. Q constraint satisfaction problem is NP-hard. We present a formal statement and proof of this connection between non-existence of polymorphisms and intractability in Appendix A.

4 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 $w = 2g+1$. In order to simplify the presentation and cleanly convey the general idea behind the reduction without having to pay too much attention to technical details, we here allow repeated literals in each clause. Proving the result without repeated literals can be done with a little more care, using the same approach as in Section 5, where we prove a stronger result for the discrepancy problem (Theorem 1.2) without allowing repeated variables in a constraint.

4.1 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 4.1. *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 4.2. *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 noted in the previous section, 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 4.3. *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 this is allowed at the moment. 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 4.2.

Proof of Lemma 4.2. 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 4.3, $f(z) = 1$ and $f(\neg z) = 1$, contradicting that f is odd. □

4.2 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 4.4 (Completeness)**.** *If* $\mathsf{Opt}(\Psi) = 1$ *then* Φ *is strongly* g*-satisfiable.*

We turn to the more interesting case of soundness.

Lemma 4.5 (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 4.2) 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 4.6. *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 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 4.5. 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 hardness part of 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.

4.3 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 4.7. *Let $g \geqslant 1$ be an integer. Assuming the Unique Games conjecture [10], 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 4.7. 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 [7], 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 [7, Thm 5.2]) shows that every pair of bits are uniformly distributed under μ. The existence of μ together with Theorem 1.3 of [2] (the hardness part of which is based on [3]) 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. □

5 Discrepancy With Small Sets

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 the previous section, though some modifications in the constructions are needed since "true" and "false" are now treated symmetrically. Additionally, we shall now deal with the details for the case when repeated elements are not allowed.

Another distinction from $(1, g, 2g + 1)$-SAT to g-DISCREPANCY is that we no longer have the concept of negated literals and so we can no longer assume that our long codes are folded. If we allowed 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 any x. Since we do not allow repetitions we have to be slightly more careful.

5.1 Dictatorship test

Let us start with the dictatorship test for a function $f : \{0, 1\}^n \to \{0, 1\}$. This consists of all sets of $2g + 1$ (distinct) variables $\{f(x_1), \ldots, f(x_{2g+1})\}$ such that for each coordinate $j \in [n]$ we have $g \leqslant \sum_{i=1}^{2g+1} x_{i,j} \leqslant g + 1$. We refer to this as the discrepancy dictatorship gadget.

The completeness of the gadget follows by construction.

Lemma 5.2. *If f is a dictatorship function then it evenly splits all sets in the discrepancy dictatorship gadget.*

For the soundness, we can no longer conclude that if f does not leave any set of the gadget monochromatic then f only depends on a few variables. For instance, if we let f equal a dictator except at a few points then f depends on all variables but would split all sets in the gadget. We prove that this is the only problem, and as a first step we start with a definition.

Definition 5.3. A function f is said to t-depend on variable i if there are at least t disjoint pairs $\{x_j, x_j \oplus e_i\}$, $j \in [t]$, such that $f(x_j) \neq f(x_j \oplus e_i)$.

Using this definition, we have the following analogue of Lemma 4.2.

Lemma 5.4. *Any function $f : \{0, 1\}^n \to \{0, 1\}$ which does not leave any set of the discrepancy dictatorship gadget monochromatic $2g$-depends on at most $2g - 1$ variables, provided $n > 5g$.*

Towards proving this, we first note that even though we can not force f to be folded by means of negated literals, the dictatorship test forces f to be folded on all but a small number of bad inputs.

Lemma 5.5. *If f does not make any set of the discrepancy dictatorship gadget monochromatic then there are at most g pairs of inputs $\{x_i, \neg x_i\}$ such that $f(x_i) = f(\neg x_i) = 0$, and at most g pairs $\{y_i, \neg y_i\}$ such that $f(y_i) = f(\neg y_i) = 1$.*

Proof. Suppose for contradiction that we have $g+1$ distinct pairs $\{x_1, \neg x_1\}, \ldots, \{x_{g+1}, \neg x_{g+1}\}$ such that $f(x_i) = f(\neg x_i) = 0$. Then f leaves $(x_1, \neg x_1, \ldots, x_g, \neg x_g, x_{g+1})$ monochromatic and it straightforward to see that this is one of allowed tests. The other claim is analogous. □

Next we have the following analogue of Claim 4.3, saying that the value of f can be essentially fixed by some setting of g variables.

Claim 5.6. *Suppose f $2g$-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 but at most $2g$ inputs z such that $z_{i_j} = c_j$ for all $j \in [g]$.*

Proof. We proceed as in the proof of Claim 4.3. For each $j \in [g]$ there are $2g$ different inputs x satisfying $f(x) = 1$ and $f(x \oplus e_{i_j}) = 0$. By Lemma 5.5, f is not folded on at most g of these and in particular there are g different choices of x such that $f(\neg x) = f(x \oplus e_{i_j}) = 0$.

It follows that we can choose g distinct inputs $x_1, \ldots, x_g$ such that $f(\neg x_j) = f(x_j \oplus e_{i_j}) = 0$ for each $j \in [g]$. Now let $c_j = x_{j,i_j}$. Consider any input z such that $z_{i_j} = c_j$ for each $j \in [g]$ but distinct from the $2g$ inputs $\neg x_1, x_1 \oplus e_{i_1}, \ldots, x_g, x_g \oplus e_{i_g}$. We claim that $\{f(z), f(\neg x_1), f(x_1 \oplus e_{i_1}), \ldots, f(\neg x_g), f(x_g \oplus e_{i_g})\}$ is a set in the discrepancy dictatorship gadget and hence $f(z)$ must equal 1.

To see this, note that in coordinate i_j the inputs $\neg x_1, x_1 \oplus e_{i_1}, \ldots, \neg x_g, x_g \oplus e_{i_g}$ have $g-1$ occurrences of the value x_{j,i_j} and $g+1$ occurrences of the value $\neg x_{j,i_j}$, and z provides one additional occurrence of the value $c_j = x_{j,i_j}$ so that this coordinate is balanced. For coordinates outside $i_1, \ldots, i_g$ the x's provide an equal number of 0's and 1's and hence the value of z in these coordinates is irrelevant. □

Proof of Lemma 5.4. The only difference to the proof of Lemma 4.2 is that we need to make sure to avoid the up to $4g$ bad inputs where f is not folded and the at most $2g + 2g$ choices of z where Claim 5.6 does not apply, but if $2^{n-2g} > 8g$ such a contradictory z exists. □

5.2 Reduction from Label Cover

The reduction from Label Cover also proceeds as in Section 4.2. The main crux is to locate a set consisting of disjoint variables. To simplify the presentation, we alter the reduction somewhat by augmenting each f_u and f_v with a set A of auxiliary variables, i.e., $f_u : \{0,1\}^{L_U \cup A} \to \{0,1\}$ and $f_v : \{0,1\}^{L_V \cup A} \to \{0,1\}$. Loosely speaking this is like creating $2^{|A|}$ copies of each variable and ensures that there is an abundance of variables allowing us to easily identify sets without repeated variables. This is technically not needed – provided the label sets L_U and L_V are sufficiently large one can find sufficiently many distinct variables in the original reduction.

In the reduction from a label cover instance Ψ, we add the discrepancy dictatorship gadget on each f_u and f_v (without any special treatment of the auxiliary variables). In the edge constraints we for an edge $e = (u, v)$ the clause

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

provided $g \leqslant \sum_{j=1}^{g} x_{j,\pi_e(i)} + \sum_{j=1}^{g+1} y_{j,i} \leqslant g+1$ for each $i \in L_v$ and the x_i's and y_i's are distinct. Note that we do not have any constraint on the sum of values in the auxiliary coordinates. Calling the resulting g-DISCREPANCY instance Φ, we have the following analogue of Lemma 4.5.

Lemma 5.7. *If there is an assignment to Φ which does not leave any set monochromatic then* $\mathsf{Opt}(\Psi) \geqslant 1/(2g-1)^2$*, provided* $|A| = \Omega(\log |L_V|)$.

Proof. As in the proof of Lemma 4.5 we take a satisfying assignment $\{f_u\}$, $\{f_v\}$ to Φ.

We now choose $S_u \subseteq L_U$ (resp. $S_v \subseteq L_V$) to be the set of *non-auxiliary* variables that f_u (resp. f_v) $4g$-depends on, which by Lemma 5.4 are at most $2g-1$ variables.

Proceeding as in Lemma 4.5, we now want to re-prove Claim 4.6, showing that S_u intersects $\pi_e(S_v)$ for the new choices of S_u and S_v. Looking at the proof of Claim 4.6, the new aspects that we need to deal with is that the inputs $x_1, \ldots, x_g, y_1, \ldots, y_{g+1}$ picked there should be distinct, and we also have the added complication that f_u and f_v may depend on variables outside S_u and S_v (though only on a very small number of input strings).

Say that a partial assignment $z \in \{0,1\}^{S_u \cup A}$ is good if:

1. $z_j = b$ for all $j \in S_u$ and some $b \in \{0,1\}$ (i.e., z is constant on the coordinates of S_u)
2. f_u is odd on all inputs agreeing with z
3. $f_u(y) = 0$ for all $y \in \{0,1\}^{L_U \cup A}$ such that $y_{|S_u \cup A} = z$.

Similarly we say that a partial assignment $w \in \{0,1\}^{S_v \cup A}$ is good provided the same conditions hold with S_u and f_u replaced by S_v and f_v.

There are $2^{|A|+1}$ partial assignments satisfying the first condition, and by Lemma 5.5 at most g partial assignments violating the second condition, so we have $2^{|A|+1} - g$ partial assignments satisfying the first two conditions.

Since f_u does not $4g$-depend on any $i \notin S_u \cup A$, there are at most $8g|L_U|$ inputs $x \in \{0,1\}^{L_U \cup A}$ for which $f(x) \neq f(x \oplus e_i)$ for some $i \notin S_u \cup A$. This implies that there are $2^{|A|+1} - g - 8g|L_U|$ partial assignments that satisfy the first two conditions and f_u restricted to z is constant. Since we have excluded the points where f_u is not odd, f_u takes the value 0 on half of the remaining assignments, so there are at least $2^{|A|} - (g + 8g|L_U|)/2$ good partial assignments z.

Similarly there are at least $2^{|A|} - (g + 8g|L_V|)/2$ good partial assignments w. It follows that if $|A| = \Omega(\log |L_V|)$ we can choose $2g+1$ distinct good partial assignments $z_1, \ldots, z_g$, and $w_1, \ldots, w_{g+1}$. See also Figure 1.

Any way of completing these partial assignments to inputs $x_1, \ldots, x_g$, $y_1, \ldots, y_{g+1}$ such that $\{f_u(x_1), \ldots, f_u(x_g), f_v(y_1), \ldots, f_v(y_{g+1})\}$ is a set appearing in Φ gives the desired contradiction since f_u (resp. f_v) is 0 on the x_j's (resp. y_j's) by construction.

This is easily done: we simply need to make sure that the number of ones in each column is g or $g+1$. The perhaps only subtle point is that we need to take care with the coordinates in $j \in \pi_e^{-1}(\pi_e(S_v))$ (i.e., the j that collide with some $j' \in S_v$ under π_e). This is where we use that the partial assignments w_i are constant on S_v, as we can then use those constant values for the remaining coordinates in $\pi_e^{-1}(\pi_e(S_v))$.

This concludes the proof that S_u intersects $\pi_e(S_v)$ for each edge (u,v), and the rest of the proof proceeds identically to Lemma 4.5. □

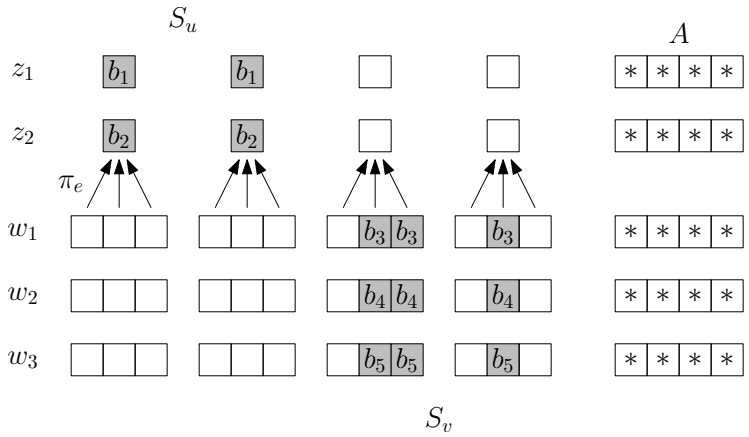

Figure 1: The good partial assignments $z_1, \ldots, z_g, w_1, \ldots, w_{g+1}$

6 Application to hereditary matrix discrepancy

Consider the following problem. Given a matrix $A = (a_{ij})$ $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$ of dimension $n \times m$, the task is to find signs $x_i \in \{-1, 1\}$, to make any subset of the columns balanced. To be more precise we want for any $\alpha \subseteq [n]$, $\alpha \neq \emptyset$, the vector $b_j^\alpha = \sum_{i \in \alpha} x_i a_{ij}$ to have small L^∞-norm. The minimum such quantity is called the *hereditary discrepancy*.

This problem was considered in [11] 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 [11] and here actually gives a matrix such that either the discrepancy of A is large (in our case $2g + 1$) or the hereditary discrepancy is small (in our case $g + 1$).

To see this let A be the incidence matrix of the set system constructed to prove Theorem 5.1. In other words $a_{ij} = 1$ if the ith element appears in set j. A setting of the x_i corresponding to a very balanced 2-coloring gives a witness that the hereditary discrepancy is at most $g + 1$ while a monochromatic set gives that the discrepancy of A is at least $2g + 1$.

7 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.

7.1 A randomized algorithm

Let us first describe a simple randomized algorithm closely following Papadimitriou's algorithm [12] for 2-Sat.

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

Proposition 7.1. *If Φ is strongly g-satisfying and $w \leqslant 2g$ then Algorithm 1 finds a satisfying assignment in $O(tn^2)$ steps with probability at least $1 - 2^{-t}$.*

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

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. Such a walk 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.

7.2 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 seem like a not 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.

Proposition 7.2. *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

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 **then return** "Not strongly g-satisfiable"
(5) Let $y_1, \ldots, y_n$ be the LP solution when y_i is forced to b
(6) **foreach** i such that $y_i \neq \frac{1}{2}$
(7) Assign $x_i \leftarrow \lfloor y_i \rceil$
(8) Remove all satisfied clauses from the formula
(9) **until** all variables are assigned
(10) **return** x

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. □

8 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 improved results quantitatively in many situations. We gave a rather modest example in Section 6 but there should be many possibilities.

As we show in Appendix A, 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. It will be extremely interesting to obtain some 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 know in the Boolean case that the existence of non-trivial polymorphisms precisely governs the tractability of the associated CSP.)

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 4.3, 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.

Acknowledgment

We are indebted to Dominik Scheder for the claim about the non-existence of gadget reductions described in Section 3.

References

[1] 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. 5

[2] Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Transactions on Computation Theory*, 5:1–24, 2013. 2, 9

[3] Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18:249–271, 2009. 9

[4] 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. 18

[5] 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. 3

[6] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1), 2009. 18

[7] Mahdi Cheraghchi, Johan Håstad, Marcus Isaksson, and Ola Svensson. Approximating linear threshold predicates. *TOCT*, 4(1):2, 2012. 9

[8] Uri Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998. 19

[9] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001. 3

[10] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of 34th ACM Symposium on Theory of Computating*, pages 767–775, 2002. 9

[11] 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 Symposium on theory of computing*, STOC '13, pages 351–360, New York, NY, USA, 2013. ACM. 13

[12] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *Proceedings of the 32nd annual symposium on Foundations of computer science*, SFCS '91, pages 163–169, Washington, DC, USA, 1991. IEEE Computer Society. 2, 13

[13] Ran Raz. A parallel repetition theorem. *SIAM J. on Computing*, 27:763–803, 1998. 5

[14] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth annual ACM Symposium on Theory of Computing*, pages 216–226, 1978. 18

A Hardness based on absence of polymorphisms

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 is thus a particular case based on the fact (established in Lemma 4.2) that there is no weak polymorphism of arity larger than $2g$ that takes assignments strongly g-satisfying a width $(2g + 1)$-clause and outputs an assignment that satisfies the clause.

We present our result for constraint satisfaction problems over an arbitrary finite domain, allowing for both folding (the generalization of "negated" variables to non-Boolean domains) and repeated literals (a "literal" associated with variable v is given by $v + a$ for some constant a in the domain, where the addition is modulo the domain size). It is not clear whether or not these two assumptions are essential or whether they can be removed as in the case of $(1, g, 2g + 1)$-SAT.

Definition A.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.

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

Definition A.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 $\mathsf{C}SP(P)$, 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 A.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)[1] 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.

In the sequel, 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.

We now state our main theorem (Theorem A.5 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 [4] that precisely ties the tractability of a CSP to the existence of non-trivial polymorphisms. Establishing the converse of Theorem A.5 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 [14] (see the survey [6] for a modern algebraic treatment of Schaefer's dichotomy theorem).

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

Definition A.4. 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 A.5 (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.*

A.1 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 A.6. *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).*

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

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 4.2:

Lemma A.7. *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.*

A.2 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 [8] for his tight inapproximability result for set cover.

Definition A.8 (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 [8].

Theorem A.9. *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 A.1 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 A.10. *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 A.11 below. Note that by picking the soundness ε of the Label Cover instance to be $\ll 1/B^2$, Theorem A.5 would follow from Theorem A.9, and Lemmas A.10 and A.11.

Lemma A.11. *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 pass 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. (This is analogous to the contradiction we obtained in the proof of Lemma 5.7.)

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). □

ECCC ISSN 1433-8092
http://eccc.hpi-web.de