Electronic Colloquium on Computational Complexity, Report No. 82 (2015)

Beating the random assignment on constraint satisfaction problems of bounded degree

Boaz Barak[*] Ankur Moitra[†] Ryan O'Donnell[‡] Prasad Raghavendra[§]
Oded Regev[¶] David Steurer[‖] Luca Trevisan[§] Aravindan Vijayaraghavan[**]
David Witmer[‡] John Wright[‡]

May 12, 2015

Abstract

We show that for any odd k and any instance $\Im$ of the Max-kXOR constraint satisfaction problem, there is an efficient algorithm that finds an assignment satisfying at least a $\frac{1}{2} + \Omega(1/\sqrt{D})$ fraction of $\Im$'s constraints, where D is a bound on the number of constraints that each variable occurs in. This improves both qualitatively and quantitatively on the recent work of Farhi, Goldstone, and Gutmann (2014), which gave a *quantum* algorithm to find an assignment satisfying a $\frac{1}{2} + \Omega(D^{-3/4})$ fraction of the equations.

For arbitrary constraint satisfaction problems, we give a similar result for "triangle-free" instances; i.e., an efficient algorithm that finds an assignment satisfying at least a $\mu + \Omega(1/\sqrt{D})$ fraction of constraints, where μ is the fraction that would be satisfied by a uniformly random assignment.

[*]Microsoft Research New England.
[†]MIT Mathematics Department.
[‡]Department of Computer Science, Carnegie Mellon.
[§]U.C.Berkeley, Department of Electrical Engineering & Computer Sciences.
[¶]Courant Institute of Mathematical Sciences, New York University.
[‖]Cornell University.
[**]Courant Institute of Mathematical Sciences, New York University.

ISSN 1433-8092

1 Introduction

An instance of a Boolean constraint satisfaction problem (CSP) over n variables $x_1, \dots, x_n$ is a collection of *constraints*, each of which is some predicate P applied to a constant number of the variables. The computational task is to find an assignment to the variables that maximizes the number of satisfied predicates. In general the constraint predicates do not need to be of the same "form"; however, it is common to study CSPs where this is the case. Typical examples include: Max-kSAT, where each predicate is the OR of k variables or their negations; Max-kXOR, where each predicate is the XOR of exactly k variables or their negations; and Max-Cut, the special case of Max-2XOR in which each constraint is of the form $x_i \neq x_j$. The case of Max-kXOR is particularly mathematically natural, as it is equivalent to maximizing a homogenous degree-k multilinear polynomial over $\{\pm 1\}^n$.

Given a CSP instance, it is easy to compute the expected fraction μ of constraints satisfied by a uniformly random assignment; e.g., in the case of Max-kXOR we always have $\mu = \frac{1}{2}$. Thus the question of algorithmic interest is to find an assignment that satisfies noticeably more than a μ fraction of constraints. Of course, sometimes this is simply not possible; e.g., for Max-Cut on the complete n-variable graph, at most a $\frac{1}{2} + O(1/n)$ fraction of constraints can be satisfied.[1] However, even when all or almost all constraints can be satisfied, it may still be algorithmically difficult to beat μ. For example, Håstad [Hås01] famously proved that for every $\varepsilon > 0$, given a Max-3XOR instance in which a $1 - \varepsilon$ fraction of constraints can be satisfied, it is NP-hard to find an assignment satisfying a $\frac{1}{2} + \varepsilon$ fraction of the constraints. Håstad showed similar "approximation resistance" results for Max-3Sat and several other kinds of CSPs.

One possible reaction to these results is to consider subconstant ε. For example, Håstad and Venkatesh [HV04] showed that for every Max-kXOR instance with m constraints, one can efficiently find an assignment satisfying at least a $\frac{1}{2} + \Omega(1/\sqrt{m})$ fraction of them.[2] (Here, and elsewhere in this introduction, the $\Omega(\cdot)$ hides a dependence on k, typically exponential.) Relatedly, Khot and Naor [KN08] give an efficient algorithm for Max-3XOR that satisfies a $\frac{1}{2} + \Omega(\varepsilon\sqrt{(\log n)/n})$ fraction of constraints whenever the optimum fraction is $\frac{1}{2} + \varepsilon$.

Another reaction to approximation resistance is to consider restricted instances. One commonly studied restriction is to assume that each variable's "degree" — i.e., the number of constraints in which it occurs — is bounded by some D. Håstad [Hås00] showed that such instances are never approximation resistant. More precisely, he showed that for, say, Max-kXOR, one can always efficiently find an assignment satisfying at least a $\mu + \Omega(1/D)$ fraction of constraints.[3] Note that this advantage of $\Omega(1/D)$ cannot in general be improved, as the case of Max-Cut on the complete graph shows.

One may also consider further structural restrictions on instances. One such restriction is that the underlying constraint hypergraph be *triangle-free* (see Section 2 for a precise definition). For example, Shearer [She92] showed that for triangle-free graphs there is an efficient algorithm for finding a cut of size at least $\frac{m}{2} + \Omega(1) \cdot \sum_i \sqrt{\deg(i)}$, where $\deg(i)$ is the degree of the ith vertex. As $\sum_i \sqrt{\deg(i)} \geq \sum_i \frac{\deg(i)}{\sqrt{D}} = \frac{2m}{\sqrt{D}}$ in m-edge degree-D bounded graphs, this shows that for *triangle-free* Max-Cut one can efficiently satisfy at least a $\frac{1}{2} + \Omega(1/\sqrt{D})$ fraction of constraints. (See [Alo97] for

[1] Another trivial example is the Max-2XOR instance with the two constraints $x = y$ and $x \neq y$. For this reason we always assume that our Max-kXOR instances do not contain a constraint and its negation.

[2] In [HV04] this is stated as an approximation-ratio guarantee: if the optimum fraction is $\frac{1}{2} + \varepsilon$ then $\frac{1}{2} + \Omega(\varepsilon/\sqrt{m})$ is guaranteed. However inspecting their proof yields the absolute statement we have made.

[3] The previous footnote applies also to this result.

a related result on Min-Bisection in degree-bounded graphs.)

1.1 Recent developments and our work

In a recent surprising development, Farhi, Goldstone, and Gutmann [FGG14] gave an efficient *quantum* algorithm that, for Max-3XOR instances with degree bound D, finds an assignment satisfying a $\frac{1}{2} + \Omega(D^{-3/4})$ fraction of the constraints. In addition, Farhi et al. show that if the Max-3XOR instance is "triangle-free" then an efficient quantum algorithm can satisfy a $\frac{1}{2} + \Omega(1/\sqrt{D})$ fraction of the constraints.

Farhi et al.'s result was perhaps the first example of a quantum algorithm providing a better CSP approximation guarantee than that of the best known classical algorithm (namely Håstad's [Hås00], for Max-3XOR). As such it attracted quite some attention.[4] In this paper we show that classical algorithms can match, and in fact outperform, Farhi et al.'s quantum algorithm.

First result: Max-kXOR. We will present two results. The first result is about instances of Max-kXOR.

Theorem 1.1. *There is a constant $c = \exp(-O(k))$ and a randomized algorithm running in time $\mathrm{poly}(m, n, \exp(k))$ that, given an instance $\mathfrak{I}$ of Max-kXOR with m constraints and degree at most D, finds with high probability an assignment $x \in \{\pm 1\}^n$ such that*

$$\left| val_{\mathfrak{I}}(x) - \frac{1}{2} \right| \geq \frac{c}{\sqrt{D}} \, . \tag{1.1}$$

Here $val_{\mathfrak{I}}(x)$ denotes the fraction of constraints satisfied by x. In particular, for odd k, by trying the assignment and its negation, the algorithm can output an x satisfying

$$val_{\mathfrak{I}}(x) \geq \frac{1}{2} + \frac{c}{\sqrt{D}} \, . \tag{1.2}$$

In Section 3 we give a simple, self-contained proof of Theorem 1.1 in the special case of Max-3XOR. For higher k we obtain it from a more general result (Theorem 4.2) that gives a constructive version of a theorem of Dinur, Friedgut, Kindler and O'Donnell [DFKO07]. This result shows how to attain a significant deviation from the random assignment value for multivariate low-degree polynomials with low influences. See Section 4.

We note that the deviation $\Omega(1/\sqrt{D})$ in (1.1) is optimal. To see why, consider any D-regular graph on n vertices, and construct a Max-2XOR instance $\mathfrak{I}$ as follows. For every edge (i, j) in the graph we randomly and independently include either the constraint $x_i = x_j$ or $x_i \neq x_j$. For every fixed x, the quantity $\mathrm{val}_{\mathfrak{I}}(x)$ has distribution $\frac{1}{m}\mathrm{Binomial}(m, \frac{1}{2})$, where $m = \frac{nD}{2}$. Hence a Chernoff-and-union-bound argument shows that with high probability all 2^n assignments will have $|\mathrm{val}_{\mathfrak{I}}(x) - \frac{1}{2}| \leq O(\sqrt{n/m}) = O(1/\sqrt{D})$. This can easily be extended to Max-kXOR for $k > 2$.

General CSPs. As noted earlier, the case of Max-Cut on the complete graph shows that for general CSPs, and in particular for Max-2XOR, we cannot guarantee a positive advantage of $\Omega(1/\sqrt{D})$ as in (1.2). In fact, a positive advantage of $\Omega(1/D)$ is the best possible, showing that the guarantee of Håstad [Hås00] is tight in general.

[4] As evidenced by the long list of authors on this paper; see also `http://www.scottaaronson.com/blog/?p=2155`.

A similar example can be shown for Max-2SAT: consider an instance with D^2 variables and imagine them placed on a $D \times D$ grid. For any two variables in the same row add the constraint $x \vee y$ and for any two variables in the same column add the constraint $\bar{x} \vee \bar{y}$. Then each variable participates in $O(D)$ clauses, and it can be verified that the best assignment satisfies $3/4 + O(1/D)$ fraction of the clauses. We do not know if the same holds for Max-3SAT and we leave that as an open question.

Sometimes no advantage over random is possible. For instance, consider the following instance with 8 clauses on 6 variables, in which *any* assignment satisfies exactly $1/2$ of the clauses:

$$\begin{aligned}\{&\mathrm{NAE}(x_1,x_2,x_3),\\ &\mathrm{AE}(y_1,x_2,x_3),\mathrm{AE}(x_1,y_2,x_3),\mathrm{AE}(x_1,x_2,y_3),\\ &\mathrm{NAE}(x_1,y_2,y_3),\mathrm{NAE}(y_1,x_2,y_3),\mathrm{NAE}(y_1,y_2,x_3),\\ &\mathrm{AE}(y_1,y_2,y_3)\}\ ,\end{aligned}$$

where NAE denotes the "not all equal" constraint, and AE is the "all equal" constraint.

Second result: triangle-free instances of general CSPs. Despite the above examples, our second result shows that it is possible to recover the optimal advantage of $1/\sqrt{D}$ for triangle-free instances of *any CSP*:

Theorem 1.2. *There is a constant* $c = \exp(-O(k))$ *and a randomized algorithm running in time* $\mathrm{poly}(m, n, \exp(k))$ *time that, given a triangle-free, degree-D CSP instance $\mathfrak{I}$ with m arbitrary constraints, each of arity between* 2 *and k, finds with high probability an assignment* $x \in \{\pm 1\}^n$ *such that*

$$val_{\mathfrak{I}}(x) \geq \mu + \frac{c}{\sqrt{D}}.$$

Here μ is the fraction of constraints in $\mathfrak{I}$ that would be satisfied in expectation by a random assignment.

This theorem is proved in Section 5. For simplicity, we state our results as achieving *randomized* algorithms and leave the question of derandomizing them (e.g., by replacing true random bits with $O(k)$-wise independence or some other such distribution) to future work.

1.2 Overview of our techniques

All three algorithms that we present in this work follow the same broad outline, while the details are different in each case. To produce an assignment that beats a random assignment, the idea is to partition the variables in to two sets (F, G) with F standing for 'Fixed' and G standing for 'Greedy' (in Section 4, these correspond to $[n] \setminus U$ and U respectively). The variables in F are assigned independent and uniform random bits and the variables in G are assigned values *greedily* based on the values already assigned to F. We will refer to constraints with exactly one variable from G as *active* constraints. The design of the *greedy* assignments and their analysis is driven by two key objectives.

1. Obtain a significant advantage over the random assignment on *active* constraints.

2. Achieve a value that is at least as good as the random assignment on inactive constraints.

The simplest example is the algorithm for Max-3XOR that we present in Section 3. First, we appeal to a *decoupling* trick due to Khot-Naor [KN08] to give an efficient approximation-preserving reduction from an arbitrary instance $\mathfrak{I}$ of Max-3XOR to a bipartite instance $\tilde{\mathfrak{I}}$. Specifically, the instance $\tilde{\mathfrak{I}}$ will contain two sets of variables $\{y_i\}_{i\in[n]}$ and $\{z_i\}_{i\in[n]}$, with every constraint having exactly one variable from $\{y_i\}_{i\in[n]}$ and two variables from $\{z_j\}_{j\in[n]}$. Notice that if we set $G = \{y_i\}_{i\in[n]}$, then objective (2) holds vacuously, i.e., every constraint in $\tilde{\mathfrak{I}}$ is active. The former objective (1) is achieved as a direct consequence of anticoncentration of low degree polynomials (see Fact 2.3). In the case of Max-kXOR, the second objective is achieved by slightly modifying the greedy assignment: we flip each of the assignments for the greedy variables with a small probability η (that corresponds to one of the extrema of the degree-k Chebyshev polynomials of the first kind).

Our algorithm for *triangle-free* instances begins by picking (F, G) to be a random partition of the variables. In this case, after fixing a random assignment to F, a natural greedy strategy would proceed as follows: Assign each variable in G a value that satisfies the maximum the number of its own active constraints.

In order to achieve objective (2), it is sufficient if for each inactive constraint its variables are assigned is independently and uniformly randomly. Since the instance is *triangle-free*, for every pair of variables $x_i, x_j \in G$ the active constraints of x_i and x_j are over disjoint sets of variables. This implies that the greedy assignments for variables within each inactive constraint are already independent. Unfortunately, the greedy assignment as defined above could possibly be biased, and in general much worse than a random assignment on the inactive constraints. We overcome this technical hurdle by using a modified greedy strategy defined as follows. Assign -1 to all variables in G and then for each variable $x_i \in G$, consider the change in the number of active constraints satisfied if we flip x_i from -1 to 1. The algorithm will flip the value only if this number exceeds an appropriately chosen threshold θ_i. The threshold θ_i is chosen so as to ensure that over all choices of values to F, the assignment to x_i is unbiased. Triangle-freeness implies that these assignments are independent within each inactive constraint. Putting these ideas together, we obtain the algorithm for triangle-free instances discussed in Section 5.

2 Preliminaries

Constraint satisfaction problems. We will be considering a somewhat general form of constraint satisfaction problems. An instance for us will consist of n Boolean variables and m constraints. We call the variables $x_1, \dots, x_n$, and we henceforth think of them as taking the Boolean values ± 1. Each constraint is a pair (P_ℓ, S_ℓ) (for $\ell \in [m]$) where $P_\ell : \{\pm 1\}^r \to \{0, 1\}$ is the *predicate*, and S_ℓ is the *scope*, an ordered r-tuple of distinct coordinates from $[n]$. The associated constraint is that $P_\ell(x_{S_\ell}) = 1$, where we use the notation x_S to denote variables x restricted to coordinates S. We always assume (without loss of generality) that P_ℓ depends on all r coordinates. The number r is called the *arity* of the constraint, and throughout this paper k will denote an upper bound on the arity of all constraints. Typically we think of k as a small constant.

We are also interested in the special case of Max-kXOR. By this we mean the case when all constraints are XORs of exactly k variables or their negations; in other words, when every P_ℓ is of the form $P_\ell(x_1, \dots, x_k) = \frac{1}{2} \pm \frac{1}{2} x_1 x_2 \cdots x_k$. When discussing Max-$k$XOR we will also always make the assumption that all scopes are distinct as sets; i.e., we don't have the same constraint or its negation more than once.

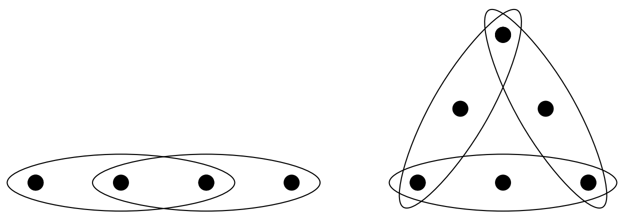

Figure 1: The two forbidden configurations for triangle-free instances

Hypergraph structure. We will be particularly interested in the *degree* $\deg(i)$ of each variable x_i in an instance. This is simply the number of constraints in which x_i participates; i.e., $\#\{\ell : S_\ell \ni i\}$. Throughout this work, we let D denote an upper bound on the degree of all variables.

For our second theorem, we will need to define the notion of "triangle-freeness".

Definition 2.1. We say that an instance is *triangle-free* if the scopes of any two distinct constraints intersect on at most one variable ("no overlapping constraints") and, moreover, there are no three distinct constraints any two of whose scopes intersect ("no hyper-triangles"), see Figure 1.

Fourier representation. We recall that any Boolean function $f : \{\pm 1\}^n \to \mathbb{R}$ can be represented by a multilinear polynomial, or *Fourier expansion*,

$$f(x) = \sum_{S \subset [n]} \widehat{f}(S) x^S, \quad \text{where } x^S \overset{\text{def}}{=} \prod_{i \in S} x_i.$$

For more details see, e.g., [O'D14]; we recall here just a few facts we'll need. First, $\mathbb{E}[f(\boldsymbol{x})] = \widehat{f}(\emptyset)$. (Here and throughout we use **boldface** for random variables; furthermore, unless otherwise specified $\boldsymbol{x}$ refers to a uniformly random Boolean string.) Second, Parseval's identity is $\|f\|_2^2 = \mathbb{E}[f(\boldsymbol{x})^2] = \sum_S \widehat{f}(S)^2$, from which it follows that $\mathrm{Var}[f(\boldsymbol{x})] = \sum_{S \neq \emptyset} \widehat{f}(S)^2$. Third,

$$\mathrm{Inf}_i[f] = \sum_{S \ni i} \widehat{f}(S)^2 = \mathbb{E}[(\partial_i f)(\boldsymbol{x})^2],$$

where $\partial_i f$ is the *derivative* of f with respect to the ith coordinate. This can be defined by the factorization $f(x) = x_i \cdot (\partial_i f)(x') + g(x')$, where $x' = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$, or equivalently by $\partial_i f(x') = \frac{f(x',+1) - f(x',-1)}{2}$, where here (x', b) denotes $(x_1, \dots, x_{i-1}, b, x_{i+1}, \dots, x_n)$. We record here a simple fact about these derivatives:

Lemma 2.2. *For any predicate $P : \{\pm 1\}^r \to \{0,1\}$, $r \geq 2$, we have $\mathrm{Var}[(\partial_i P)(\boldsymbol{x})] \geq \Omega(2^{-r})$ for all i.*

Proof. The function $\partial_i P(x)$ takes values in $\{-\frac{1}{2}, 0, \frac{1}{2}\}$. It cannot be constantly 0, since we assume P depends on its ith input. It also cannot be constantly $\frac{1}{2}$, else we would have $P(x) = \frac{1}{2} + \frac{1}{2}x_i$ and so P would not depend on all $r \geq 2$ coordinates. Similarly it cannot be constantly $-\frac{1}{2}$. Thus $\partial_i P(x)$ is nonconstant, so its variance is $\Omega(2^{-r})$. $\square$

Given an instance and an assignment $x \in \{\pm 1\}^n$, the number of constraints satisfied by the assignment is simply $\sum_\ell P_\ell(x_{S_\ell})$. This can be thought of as a multilinear polynomial $\{\pm 1\}^n \to \mathbb{R}$ of degree[5] at most k. We would like to make two minor adjustments to it, for simplicity. First, we

[5]We have the usual unfortunate terminology clash; here we mean degree as a polynomial.

will normalize it by a factor of $\frac{1}{m}$ so as to obtain the *fraction* of satisfied constraints. Second, we will replace P_ℓ with $\overline{P}_\ell$, defined by

$$\overline{P}_\ell = P_\ell - \mathbb{E}[P_\ell] = P_\ell - \widehat{P}_\ell(\varnothing).$$

In this way, $\overline{P}_\ell(x_{S_\ell})$ represents the *advantage* over a random assignment. Thus given an instance, we define the *associated polynomial* $\mathfrak{P}(x)$ by

$$\mathfrak{P}(x) = \frac{1}{m}\sum_{\ell=1}^{m} \overline{P}_\ell(x_{S_\ell}).$$

This is a polynomial of degree at most k whose value on an assignment x represents the advantage obtained over a random assignment in terms of the fraction of constraints satisfied. In general, the algorithms in this paper are designed to find assignments $x \in \{\pm 1\}^n$ with $\mathfrak{P}(x) \geq \Omega(\frac{1}{\sqrt{D}})$.

Low-degree polynomials often achieve their expectation. Our proofs will frequently rely on the following fundamental fact from Fourier analysis, whose proof depends on the well-known "hypercontractive inequality". A proof of this fact appears in, e.g., [O'D14, Theorem 9.24].

Fact 2.3. *Let $f : \{\pm 1\}^n \to \mathbb{R}$ be a multilinear polynomial of degree at most k. Then $\mathbb{P}[f(\boldsymbol{x}) \geq \mathbb{E}[f]] \geq \frac{1}{4}\exp(-2k)$. In particular, by applying this to f^2, which has degree at most $2k$, we get*

$$\mathbb{P}\Big[|f(\boldsymbol{x})| \geq \|f\|_2\Big] \geq \exp(-O(k))$$

which implies that

$$\mathbb{E}\Big[|f(\boldsymbol{x})|\Big] \geq \exp(-O(k)) \cdot \|f\|_2 \geq \exp(-O(k)) \cdot \mathrm{stddev}[f(\boldsymbol{x})]\,.$$

3 A simple proof for Max-3XOR

We begin by proving Theorem 1.1 in the case of Max-3XOR, as the proof can be somewhat streamlined in this case. Given an instance of Max-3XOR we have the corresponding polynomial

$$\mathfrak{P}(x) = \sum_{|S|=3} \widehat{\mathfrak{P}}(S)x^S = \sum_{i,j,k\in[n]} a_{ijk}x_ix_jx_k,$$

where $\widehat{\mathfrak{P}}(S) \in \{\pm\frac{1}{2m}, 0\}$ depending on whether the corresponding constraint exists in the instance, and where we have introduced $a_{ijk} = \frac{1}{6}\widehat{\mathfrak{P}}(\{i,j,k\})$ for $i,j,k \in [n]$ distinct. We now use the trick of "decoupling" the first coordinate (cf. [KN08, Lem. 2.1]); i.e., our algorithm will consider $\widetilde{\mathfrak{P}}(y,z) = \sum_{i,j,k} a_{ijk}y_iz_jz_k$, where $y_1,\dots,y_n,z_1,\dots,z_n$ are new variables. The algorithm will ultimately produce a good assignment $y,z \in \{\pm 1\}^n$ for $\widetilde{\mathfrak{P}}$. Then it will define an assignment $\boldsymbol{x} \in \{\pm 1\}^n$ by using one of three "randomized rounding" schemes:

$$\text{w.p. } \tfrac{4}{9},\ \boldsymbol{x}_i = \begin{cases} y_i & \text{w.p. } \frac{1}{2} \\ z_i & \text{w.p. } \frac{1}{2} \end{cases} \forall i; \qquad \text{w.p. } \tfrac{4}{9},\ \boldsymbol{x}_i = \begin{cases} y_i & \text{w.p. } \frac{1}{2} \\ -z_i & \text{w.p. } \frac{1}{2} \end{cases} \forall i; \qquad \text{w.p. } \tfrac{1}{9},\ \boldsymbol{x}_i = -y_i \ \forall i.$$

We have that $\mathbb{E}[\mathfrak{P}(\boldsymbol{x})]$ is equal to

$$\tfrac{4}{9}\sum_{i,j,k} a_{ijk}(\tfrac{y_i+z_i}{2})(\tfrac{y_j+z_j}{2})(\tfrac{y_k+z_k}{2}) + \tfrac{4}{9}\sum_{i,j,k} a_{ijk}(\tfrac{y_i-z_i}{2})(\tfrac{y_j-z_j}{2})(\tfrac{y_k-z_k}{2}) + \tfrac{1}{9}\sum_{i,j,k} a_{ijk}(-y_i)(-y_j)(-y_k)$$

$$= \frac{1}{9}\sum_{i,j,k} a_{ijk}(y_i z_j z_k + z_i y_j z_k + z_i z_j y_k) = \frac{1}{3}\widetilde{\mathfrak{P}}(y,z). \tag{3.1}$$

Thus in expectation, the algorithm obtains an assignment for $\mathfrak{P}$ achieving at least $\frac{1}{3}$ of what it achieves for $\widetilde{\mathfrak{P}}$.

Let us now write $\widetilde{\mathfrak{P}}(y,z) = \sum_i y_i G_i(z)$, where $G_i(z) = \sum_{j,k} a_{ijk} z_j z_k$. It suffices for the algorithm to find an assignment for z such that $\sum_i |G_i(z)|$ is large, as it can then achieve this quantity by taking $y_i = \operatorname{sgn}(G_i(z))$. The algorithm simply chooses $\boldsymbol{z} \in \{\pm 1\}^n$ uniformly at random. By Parseval we have $\mathbb{E}[G_i(\boldsymbol{z})^2] = \sum_{j<k}(2a_{ijk})^2 = \frac{1}{9}\operatorname{Inf}_i[\mathfrak{P}]$ for each i. Applying Fact 2.3 (with $k = 2$) we therefore get $\mathbb{E}[|G_i(\boldsymbol{z})|] \geq \Omega(1)\cdot\sqrt{\operatorname{Inf}_i[\mathfrak{P}]}$. Since $\operatorname{Inf}_i[\mathfrak{P}] = \deg(i)/4m^2$, we conclude

$$\mathbb{E}\left[\sum_i |G_i(\boldsymbol{z})|\right] \geq \Omega(1)\cdot\sum_i \frac{\sqrt{\deg(i)}}{m} \geq \Omega(1)\cdot\sum_i \frac{\deg(i)}{m\sqrt{D}} = \Omega(1)\cdot\frac{1}{\sqrt{D}}.$$

As $\sum_i |G_i(\boldsymbol{z})|$ is bounded by $1/2$, Markov's inequality implies that the algorithm can with high probability find a z achieving $\sum_i |G_i(z)| \geq \Omega(\frac{1}{\sqrt{D}})$ after $O(\sqrt{D})$ trials of $\boldsymbol{z}$. As stated, the algorithm then chooses y appropriately to attain $\widetilde{\mathfrak{P}}(y,z) \geq \Omega(\frac{1}{\sqrt{D}})$, and finally gets $\frac{1}{3}$ of this value (in expectation) for $\mathfrak{P}(x)$.

Derandomization. It is easy to efficiently derandomize the above algorithm. The main step is to recognize that "$(2,4)$-hypercontractivity" is all that's needed for Fact 2.3 (perhaps with a worse constant); thus it holds even when the random bits are merely 4-wise independent. This is well known, but we could not find an explicit reference; hence we give the proof in the case when f is homogeneous of degree 2 (the case that's needed in the above algorithm). Without loss of generality we may assume $\mathbb{E}[f(\boldsymbol{x})] = 0$ and $\mathbb{E}[f(\boldsymbol{x})^2] = 1$. Then it's a simple exercise to check that $\mathbb{E}[f(\boldsymbol{x})^4] \leq 15$, and this only requires the bits of $\boldsymbol{x}$ to be 4-wise independent. But now

$$\mathbb{P}[f(\boldsymbol{x}) \geq 0] = \mathbb{E}[1_{\{f(\boldsymbol{x})\geq 0\}}] \geq \mathbb{E}[.13f(\boldsymbol{x}) + .06f(\boldsymbol{x})^2 - .002f(\boldsymbol{x})^4] \geq .06 - .002\cdot 15 = .03$$

where we used the elementary fact $1_{\{t\geq 0\}} \geq .13t + .06t^2 - .002t^4$ for all $t \in \mathbb{R}$. Thus indeed the algorithm can find a z achieving $\sum_i |G_i(z)| \geq \Omega(\frac{1}{\sqrt{D}})$ by enumerating all strings in a 4-wise independent set; it is well known this can be done in polynomial time. Following this, the algorithm chooses string y deterministically. Finally, it is clear that each of the three different randomized rounding schemes only requires 3-wise independence, and a deterministic algorithm can simply try all three and choose the best one.

4 A general result for bounded-influence functions

One can obtain our Theorem 1.1 for higher odd k by generalizing the proof in the preceding section. Constructing the appropriate "randomized rounding" scheme to decouple the first variable becomes slightly more tricky, but one can obtain the identity analogous to (3.1) through the use of Chebyshev polynomials. At this point the solution becomes very reminiscent of the Dinur et al. [DFKO07] work. Hence in this section we will simply directly describe how one can make [DFKO07] algorithmic.

The main goal of [DFKO07] was to understand the "Fourier tails" of bounded degree-k polynomials. One of their key technical results was the following theorem, showing that if a degree-k polynomial has all of its influences small, it must deviate significantly from its mean with noticeable probability:

Theorem 4.1. *([DFKO07, Theorem 3].) There is a universal constant C such that the following holds. Suppose $g : \{\pm 1\}^n \to \mathbb{R}$ is a polynomial of degree at most k and assume* $\mathrm{Var}[g] = 1$. *Let $t \geq 1$ and suppose that* $\mathrm{Inf}_i[g] \leq C^{-k} t^{-2}$ *for all $i \in [n]$. Then*

$$\mathbb{P}[|g(\boldsymbol{x})| \geq t] \geq \exp(-Ct^2 k^2 \log k).$$

In the context of Max-kXOR, this theorem already nearly proves our Theorem 1.1. The reason is that in this context, the associated polynomial $\mathfrak{P}(x)$ is given by

$$\mathfrak{P}(x) = \frac{1}{2m} \sum_{\ell=1}^{m} b_\ell \prod_{j \in S_\ell} x_j, \text{ where } b_\ell \in \{-1, 1\}.$$

Hence $\mathrm{Var}[\mathfrak{P}] = 1/4m$ and $\mathrm{Inf}_i[\mathfrak{P}] = \deg(x_i)/4m^2 \leq D/4m^2$. Taking $g = 2\sqrt{m} \cdot \mathfrak{P}$ and $t = \exp(-O(k)) \cdot \sqrt{m/D}$, Theorem 4.1 immediately implies that

$$\mathbb{P}\left[|\mathfrak{P}(\boldsymbol{x})| \geq \exp(-O(k)) \cdot \frac{1}{\sqrt{D}}\right] \geq \exp(-O(m/D)). \tag{4.1}$$

This already shows the desired existential result, that there *exists* an assignment beating the random assignment by $\exp(-O(k)) \cdot \frac{1}{\sqrt{D}}$. The only difficulty is that the low probability bound in (4.1) does not imply we can find such an assignment efficiently.

However this difficulty really only arises because [DFKO07] had different goals. In their work, it was essential to show that g achieves a slightly large value on a completely *random* input.[6] By contrast, we are at liberty to show g achieves a large value however we like — semi-randomly, greedily — so long as our method is algorithmic. That is precisely what we do in this section of the paper. Indeed, in order to "constructivize" [DFKO07], the only fundamental adjustment we need to make is at the beginning of the proof of their Lemma 1.3: when they argue that "$\mathbb{P}[|\ell(\boldsymbol{x})| \geq t'] \geq \exp(-O(t'^2))$ for the degree-1 polynomial $\ell(x)$", we can simply greedily choose an assignment x with $|\ell(x)| \geq t'$.

Our constructive version of Theorem 4.1 follows. It directly implies our Theorem 1.1, as described above.

Theorem 4.2. *There is a universal constant C and a randomized algorithm such that the following holds. Let $g : \{\pm 1\}^n \to \mathbb{R}$ be a polynomial with degree at most k and* $\mathrm{Var}[g] = 1$ *be given. Let $t \geq 1$ and suppose that* $\mathrm{Inf}_i[g] \leq C^{-k} t^{-2}$ *for all $i \in [n]$. Then with high probability the algorithm outputs an assignment x with $|g(x)| \geq t$. The running time of the algorithm is* $\mathrm{poly}(m, n, \exp(k))$, *where m is the number of nonzero monomials in g.*[7]

The algorithm AdvRand achieving Theorem 4.2 is given below. It is derived directly from [DFKO07], and succeeds with probability that is inverse polynomial in n.The success probability is then boosted by running the algorithm multiple times. We remark that $\eta_0^{(k)}, \eta_1^{(k)}, \ldots, \eta_k^{(k)}$ denote the $k+1$ extrema in $[-1, 1]$ of the kth Chebyshev polynomial of the first kind $T_k(x)$, and are given by $\eta_j^{(k)} = \cos(j\pi/k)$ for $0 \leq j \leq k$. We now describe the algorithm below, for completeness. In the rest of the section, we will assume without loss of generality that k is odd (for even k, we just think of the polynomial as being of degree $k+1$, with the degree $(k+1)$ part being 0).

[6]Also, their efforts were exclusively focused on the parameter k, with quantitative dependencies on t not mattering. Our focus is essentially the opposite.

[7]For simplicity in our algorithm, we assume that exact real arithmetic can be performed efficiently.

AdvRand: Algorithm for Advantage over Average for degree k polynomials

Input: a degree k-function g
Output: an assignment x

```
1. Let $1 \leq s \leq \log_2 k$ be a scale such that the mass (i.e., sum of squares of coefficients) of the Fourier transform of $g$ on levels between $2^{s-1}$ and $2^s$ is at least $1/\log k$.
2. For every $i \in [n]$, put $i$ in set $U$ with probability $2^{-s}$. For every $i \notin U$, set $x_i \in \{-1,1\}$ uniformly at random and let $y$ be the assignment restricted to the variables in $[n] \setminus U$.
3. Let $g_y$ be the restriction obtained. For every $j \in U$, set $x_j = \mathrm{sign}(\widehat{g}_y(\{j\}))$.
4. Pick $r \in \{0,1,\dots,k\}$ uniformly at random, and let $\eta = \eta_r^{(k)}/2$.
5. For each coordinate $j \in U$, flip $x_j$ independently at random with probability $(1-\eta)/2$.
6. Output $x$.
```

We now give the analysis of the algorithm, following [DFKO07]. The second step of the algorithm performs a *random restriction*, that ensures that g_y has a lot of mass on the first-order Fourier coefficients. The key lemma (that follows from the proof of Lemma 1.3 and Lemma 4.1 in [DFKO07]) shows that we can find an assignment that obtains a large value for a polynomial with sufficient "smeared" mass on the first-order Fourier coefficients.

Lemma 4.3. *Suppose $g : \{\pm 1\}^N \to \mathbb{R}$ has degree at most k, $t \geq 1$, and $\sum_{i \in [N]} |\widehat{g}(\{i\})| \geq 2t(k+1)$. Then a randomized polynomial time algorithm outputs a distribution over assignments $\boldsymbol{x} \in \{-1,1\}^N$ such that*

$$\mathop{\mathbb{P}}_{\boldsymbol{x}}[|g(\boldsymbol{x})| \geq t] \geq \exp(-O(k)).$$

The algorithm proving Lemma 4.3 corresponds to Steps (3-6) of the Algorithm AdvRand.

Proof. We sketch the proof, highlighting the differences to Lemma 1.3 of [DFKO07]. First we observe that by picking the assignment $x_i^* = \mathrm{sign}(\widehat{g}(\{i\}))$, we can maximize the linear portion as

$$\sum_{i \in [N]} \widehat{g}(\{i\}) x_i^* = \sum_{i \in [N]} |\widehat{g}(\{i\})| \geq 2t(k+1).$$

From this point on, we follow the proof of Lemma 1.3 in [DFKO07] with their initial point x_0 being set to x^*. Let $\boldsymbol{z} \leftarrow_\eta \{\pm 1\}^N$ be a random string generated by independently setting each coordinate $\boldsymbol{z}_j = -1$ with probability $(1-\eta)/2$ (as in step 5 of the algorithm), and let

$$(T_\eta g)(x^*) = \mathop{\mathbb{E}}_{\boldsymbol{z} \leftarrow_\eta \{\pm 1\}^n} [g(x^* \cdot \boldsymbol{z})].$$

Lemma 1.3 of [DFKO07], by considering $(T_\eta g)(x^*)$ as a polynomial in η and using the extremal properties of Chebyshev polynomials (Corollary 2.8 in [DFKO07]), shows that there exists $\eta \in \{\frac{\eta_0^{(k)}}{2}, \frac{\eta_1^{(k)}}{2}, \dots, \frac{\eta_k^{(k)}}{2}\}$ such that

$$\mathop{\mathbb{E}}_{\boldsymbol{z} \leftarrow_\eta \{\pm 1\}^n} \left[|g(x^* \cdot \boldsymbol{z})| \right] \geq 2t(k+1) \cdot \frac{1}{(2k+2)} = t. \tag{4.2}$$

Consider $g(x^* \cdot z)$ as a polynomial in z, with degree at most k. As in [DFKO07], we will now use the hypercontractivity to give a lower bound on the probability (over random z) that $|g(x^* \cdot z)|$ exceeds the expectation. Note that our choice of $\eta \in [-\frac{1}{2}, \frac{1}{2}]$ and hence the bias is in the interval $[\frac{1}{4}, \frac{3}{4}]$. Using Lemma 2.5 in [DFKO07] (the analogue of Fact 2.3 for biased measures), it follows that

$$\mathop{\mathbb{P}}_{z}\left[|g(x^* \cdot z)| \geq t\right] \geq \tfrac{1}{4}\exp(-2k).$$

Hence when x is picked according to $\mathcal{D}$, with probability at least $1/(k+1)$ the algorithm chooses an η such that (4.2) holds, and then a random z succeeds with probability $\exp(-O(k))$, thereby giving the required success probability. $\square$

We now sketch the proof of the constructive version of Theorem 3 in [DFKO07], highlighting why algorithm AdvRand works.

Proof of Theorem 4.2. The scale s is chosen such that the Fourier coefficients of g of order $[2^{s-1}, 2^s]$ have mass at least $1/\log k$. The algorithm picks set U randomly by choosing each variable with probability 2^{-s}, and g_y is the restriction of g to the coordinates in U obtained by setting the other variables randomly to $y \in \{-1,1\}^{[N]\setminus U}$.

Let $\gamma_i = \sum_{S \cap U = \{i\}} \widehat{g}(S)^2$. Fixing U and y, let the indices $T = \{i \in U : \widehat{g}_y(\{i\})^2 \leq (2e)^{2k}\gamma_i\}$. The proof of Theorem 3 in [DFKO07] shows that a constant fraction of the first order Fourier coefficients are large; in particular after Steps 1 and 2 of the algorithm,

$$\mathop{\mathbb{P}}_{U,y}\left[\sum_{i\in T}\widehat{g}_y(\{i\})^2 \geq \frac{1}{100\log k}\right] \geq \exp(-O(k))\ . \tag{4.3}$$

Further, for $i \in T$, we have $|\widehat{g}_y(\{i\})| \leq (2e)^k\sqrt{\gamma_i} \leq (2e)^k\sqrt{\mathrm{Inf}_i(g)}$. Hence, when the above event in (4.3) is satisfied we have

$$\begin{aligned}\sum_{i\in U}|\widehat{g}_y(\{i\})| &\geq \frac{1}{\max_{i\in T}|\widehat{g}_y(\{i\})|}\cdot\sum_{i\in T}\widehat{g}_y(\{i\})^2 \\ &\geq \frac{1}{(2e)^k\sqrt{\max_i \mathrm{Inf}_i(g)}}\cdot\frac{1}{100\log k} \geq 2t(k+1).\end{aligned}$$

Hence, applying Lemma 4.3 with g_y we get that

$$\mathop{\mathbb{P}}_{x\in\mathcal{D}}\left[|g(x)| \geq t\right] \geq \exp(-O(k)), \tag{4.4}$$

where $\mathcal{D}$ is the distribution over assignments x output by the algorithm. Repeating this algorithm $\exp(O(k))$ times, we get the required high probability of success. $\square$

5 Triangle-free instances

In this section we present the proof of Theorem 1.2, which gives an efficient algorithm for beating the random assignment in the case of arbitrary triangle-free CSPs (recall Definition 2.1). We now restate Theorem 1.2 and give its proof. As in the proof of Theorem 4.2, we can easily move from an expectation guarantee to a high probability guarantee by first applying Markov's inequality, and then repeating the algorithm $\exp(k)\,\mathrm{poly}(n,m)$ times; hence we will prove the expectation guarantee here.

Theorem 5.1. *There is a* $\mathrm{poly}(m, n, \exp(k))$*-time randomized algorithm with the following guarantee. Let the input be a triangle-free instance over n Boolean variables, with m arbitrary constraints each of arity between* 2 *and k. Assume that each variable participates in at most D constraints. Let the associated polynomial be* $\mathfrak{P}(x)$*. Then the algorithm outputs an assignment* $\boldsymbol{x} \in \{\pm 1\}^n$ *with*

$$\mathbb{E}[\mathfrak{P}(\boldsymbol{x})] \geq \exp(-O(k)) \cdot \sum_{i=1}^{n} \frac{\sqrt{\deg(i)}}{m} \geq \exp(-O(k)) \cdot \frac{1}{\sqrt{D}}.$$

Proof. Let (F, G) be a partition of $[n]$, with F standing for "Fixed" and G standing for "Greedy". Eventually the algorithm will choose the partition randomly, but for now we treat it as fixed. We will write the two parts of the algorithm's random assignment $\boldsymbol{x}$ as $(\boldsymbol{x}_F, \boldsymbol{x}_G)$. The bits $\boldsymbol{x}_F$ will first be chosen independently and uniformly at random. Then the bits $\boldsymbol{x}_G$ will be chosen in a careful way which will make them uniformly random, but not completely independent.

To make this more precise, define a constraint (P_ℓ, S_ℓ) to be *active* if its scope S_ℓ contains exactly one coordinate from G. Let us partition these active constraints into groups

$$N_j = \{\ell : S_\ell \text{ is active and } S_\ell \ni j\}, \quad j \in G.$$

For each coordinate $j \in G$, we'll define $A_j \subset F$ to be the union of all active scopes involving j (but excluding j itself); i.e.,

$$A_j = \bigcup \{S_\ell \setminus \{j\} : \ell \in N_j\}.$$

This set A_j may be empty. Our algorithm's choice of $\boldsymbol{x}_G$ will have the following property:

$\forall j \in G$, *the distribution of* $\boldsymbol{x}_j$ *is uniformly random, and it depends only on* $(\boldsymbol{x}_i : i \in A_j)$. (†)

From property (†) we may derive:

Claim 5.1.1. *For every* inactive *constraint* (P_ℓ, S_ℓ)*, the random assignment bits* $\boldsymbol{x}_{S_\ell}$ *are uniform and* independent.

Proof of Claim. First consider the coordinates $j \in S_\ell \cap G$. By the property (†), each such $\boldsymbol{x}_j$ depends only on $(\boldsymbol{x}_i : i \in A_j)$; further, these sets A_j are disjoint precisely because of the "no hyper-triangles" part of triangle-freeness. Thus indeed the bits $(\boldsymbol{x}_j : j \in S_\ell \cap G)$ are uniform and mutually independent. The remaining coordinates $S_\ell \cap F$ are also disjoint from all these $(A_j)_{j \in S_\ell \cap G}$, by the "no overlapping constraints" part of the triangle-free property. Thus the remaining bits $(\boldsymbol{x}_i : i \in S_\ell \cap F)$ are uniform, independent, and independent of the bits $(\boldsymbol{x}_j : j \in S_\ell \cap G)$, completing the proof of the claim. □

An immediate corollary of the claim is that all inactive constraints $\overline{P}_\ell$ contribute nothing, in expectation, to $\mathbb{E}[\mathfrak{P}(\boldsymbol{x})]$. Thus it suffices to consider the contribution of the active constraints. Our main goal will be to show that the bits $\boldsymbol{x}_G$ can be chosen in such a way that

$$\forall j \in G \quad \mathbb{E}\Big[\sum_{\ell \in N_j} \overline{P}_\ell(\boldsymbol{x}_{S_\ell})\Big] \geq \exp(-O(k)) \cdot \sqrt{|N_j|} \tag{5.1}$$

and hence

$$\mathbb{E}[\mathfrak{P}(\boldsymbol{x})] \geq \frac{1}{m} \cdot \exp(-O(k)) \cdot \sum_{j \in G} \sqrt{|N_j|}. \tag{5.2}$$

Given (5.2) it will be easy to complete the proof of the theorem by choosing the partition (F, G) randomly.

So towards showing (5.1), fix any $j \in G$. For each $\ell \in N_j$ we can write $\overline{P}_\ell(x_{S_\ell}) = x_j Q_\ell(x_{S_\ell \setminus \{j\}}) + R_\ell(x_{S_\ell \setminus \{j\}})$, where $Q_\ell = \partial_j \overline{P}_\ell = \partial_j P_\ell$. Since the bits $\boldsymbol{x}_i$ for $i \in S_\ell \setminus \{j\} \subset F$ are chosen uniformly and independently, the expected contribution to (5.1) from the R_ℓ polynomials is 0. Thus we just need to establish

$$\mathbb{E}\Big[\boldsymbol{x}_j \cdot \sum_{\ell \in N_j} \boldsymbol{Q}_\ell\Big] \geq \exp(-O(k)) \cdot \sqrt{|N_j|}, \quad \text{where } \boldsymbol{Q}_\ell \stackrel{\text{def}}{=} Q_\ell(\boldsymbol{x}_{S_\ell \setminus \{j\}}). \tag{5.3}$$

We now finally describe how the algorithm chooses the random bit $\boldsymbol{x}_j$. Naturally, we will choose it to be $+1$ when $\sum_{\ell \in N_j} \boldsymbol{Q}_\ell$ is "large" and -1 otherwise. Doing this satisfies the second aspect of property (†), that $\boldsymbol{x}_j$ should depend only on $(\boldsymbol{x}_i : i \in A_j)$. To satisfy the first aspect of property (†), that $\boldsymbol{x}_j$ is equally likely ± 1, we are essentially forced to define

$$\boldsymbol{x}_j = \operatorname{sgn}\Big(\sum_{\ell \in N_j} \boldsymbol{Q}_\ell - \theta_j\Big), \tag{5.4}$$

where θ_j is defined to be a *median* of the random variable $\sum_{\ell \in N_j} \boldsymbol{Q}_\ell$.

(Actually, we have to be a little careful about this definition. For one thing, if the median θ_j is sometimes achieved by the random variable, we would have to carefully define $\operatorname{sgn}(0)$ to be sometimes $+1$ and sometimes -1 so that $\boldsymbol{x}_j$ is equally likely ± 1. For another thing, we are assuming here that the algorithm can efficiently *compute* the medians θ_j. We will describe how to handle these issues in a technical remark after the proof.)

Having described the definition (5.4) of $\boldsymbol{x}_j$ satisfying property (†), it remains to verify the inequality (5.3). Notice that by the "no overlapping constraints" aspect of triangle-freeness, the random variables $\boldsymbol{Q}_\ell$ are actually mutually independent. Further, Lemma 2.2 implies that each has variance $\Omega(2^{-k})$; hence the variance of $\boldsymbol{Q} \stackrel{\text{def}}{=} \sum_{\ell \in N_j} \boldsymbol{Q}_\ell$ is $\exp(-O(k)) \cdot |N_j|$. Thus inequality (5.3) is equivalent to

$$\mathbb{E}[\operatorname{sgn}(\boldsymbol{Q} - \theta_j)\boldsymbol{Q}] \geq \exp(-O(k)) \cdot \operatorname{stddev}[\boldsymbol{Q}] = \exp(-O(k)) \cdot \operatorname{stddev}[\boldsymbol{Q} - \theta_j].$$

Now

$$\mathbb{E}[\operatorname{sgn}(\boldsymbol{Q} - \theta_j)\boldsymbol{Q}] = \mathbb{E}[\operatorname{sgn}(\boldsymbol{Q} - \theta_j)(\boldsymbol{Q} - \theta_j + \theta_j)] = \mathbb{E}[|\boldsymbol{Q} - \theta_j|] + \mathbb{E}[\boldsymbol{x}_j \cdot \theta_j]. \tag{5.5}$$

We have $\mathbb{E}[\boldsymbol{x}_j \cdot \theta_j] = 0$ since $\mathbb{E}[\boldsymbol{x}_j] = 0$. And as for $\mathbb{E}[|\boldsymbol{Q} - \theta_j|]$, it is indeed at least $\exp(-O(k)) \cdot \operatorname{stddev}[\boldsymbol{Q}]$ by Fact 2.3, since $\boldsymbol{Q}$ is a degree-$(k-1)$ function of uniform and independent random bits. Thus we have finally established (5.1), and therefore (5.2).

To conclude, we analyze what happens when the algorithm initially chooses a uniformly *random* partition $(\boldsymbol{F}, \boldsymbol{G})$ of $[n]$. In light of (5.2), it suffices to show that for each $i \in [n]$ we have

$$\mathbb{E}\Big[\mathbf{1}[i \in \boldsymbol{G}] \cdot \sqrt{|\boldsymbol{N}_i|}\Big] \geq \exp(-O(k)) \cdot \sqrt{\deg(i)}. \tag{5.6}$$

We have $\mathbb{P}[i \in \boldsymbol{G}] = \frac{1}{2}$; conditioning on this event, let us consider the random variable $|\boldsymbol{N}_i|$; i.e., the number of active constraints involving variable x_i. A constraint scope S_ℓ containing i becomes active if and only if all the other indices in S_ℓ go into $\boldsymbol{F}$, an event that occurs with probability 2^{-k+1} (at least). Furthermore, these events are independent across the scopes containing i because of the "no overlapping constraints" property of triangle-freeness. Thus (conditioned on $i \in \boldsymbol{G}$), each random variable $|\boldsymbol{N}_i|$ is the sum $\boldsymbol{A}_1 + \cdots + \boldsymbol{A}_{\deg(i)}$ independent indicator random variables, each with expectation at least 2^{-k+1}. Thus we indeed have $\mathbb{E}[\sqrt{|\boldsymbol{N}_i|}] \geq \exp(-O(k))\sqrt{\deg(i)}$ as needed to complete the proof of (5.6). This follows from the well known fact that $\mathbb{E}[\sqrt{\text{Binomial}(d,p)}] \geq$

$\Omega(\min(\sqrt{dp}, dp))$. (Alternatively, this follows from the fact that $\boldsymbol{A}_1 + \cdots + \boldsymbol{A}_{d_i}$ is at least its expectation $d_i 2^{-k+1}$ with probability at least $\exp(-O(k))$, by Fact 2.3. Here we would use that the $\boldsymbol{A}_j$'s are degree-$(k-1)$ functions of independent random bits defining $(\boldsymbol{F}, \boldsymbol{G})$). The proof is complete. $\square$

Remark 5.2. Regarding the issue of algorithmically obtaining the medians in the above proof: In fact, we claim it is unnecessary for the algorithm to compute the median θ_j of each $\boldsymbol{Q}_j$ precisely. Instead, our algorithm will (with high probability) compute a number $\widetilde{\theta}_j$ and a probabilistic way of defining $\operatorname{sgn}(0) \in \{\pm 1\}$ such that, when $\boldsymbol{x}_j$ is defined to be $\operatorname{sgn}(\boldsymbol{Q} - \widetilde{\theta}_j)$, we have $|\mathbb{E}[\boldsymbol{x}_j]| \le \delta$, where $\delta = 1/\operatorname{poly}(m, n, \exp(k))$ is sufficiently small. First, let us briefly say why this is sufficient. The above proof relied on $\mathbb{E}[\boldsymbol{x}_j] = 0$ in two places. One place was in the last term of (5.5), where we used $\mathbb{E}[\boldsymbol{x}_j \cdot \theta_j] = 0$. Now in the approximate case, we'll have $|\mathbb{E}[\boldsymbol{x}_j \cdot \widetilde{\theta}_j]| \le \delta m$, and by taking δ appropriately small this will contribute negligibly to the overall theorem. The other place that $\mathbb{E}[\boldsymbol{x}_j] = 0$ was used was in deducing from Claim 5.1.1, that the inactive constraints contributed nothing to the algorithm's expected value. When we merely have $|\mathbb{E}[\boldsymbol{x}_j]| \le \delta$ (but still have the independence used in the claim), it's easy to see from Fourier considerations that each inactive constraint still contributes at most $2^k\delta$ to the overall expectation, and again this is negligible for the theorem as a whole if $\delta = 1/\operatorname{poly}(m, n, \exp(k))$ is sufficiently small. Finally, it is not hard to show that the algorithm can compute an appropriate $\widetilde{\theta}_j$ and probabilistic definition of $\operatorname{sgn}(0)$ in $\operatorname{poly}(m, n, \exp(k))$ time (with high probability), just by sampling to find a good approximate median $\widetilde{\theta}_j$ and then also estimating $\mathbb{P}[\boldsymbol{Q}_j = \widetilde{\theta}_j]$ to handle the definition of $\operatorname{sgn}(0)$.

Acknowledgments

We thank Scott Aaronson for bringing the paper of Farhi et al. [FGG14] to (some of) the authors' attention. RO, DW, and JW were supported by NSF grants CCF-0747250 and CCF-1116594. DW was also supported by the NSF Graduate Research Fellowship Program under grant DGE-1252522; JW was also supported by a Simons Graduate Fellowship. OR, DS, and AV acknowledge the support of the Simons Collaboration on Algorithms and Geometry. OR was also supported by NSF grant CCF-1320188. DS was also supported by a Sloan fellowship, a Microsoft Research Faculty Fellowship, and by the NSF. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[Alo97] Noga Alon, *On the edge-expansion of graphs*, Combin. Probab. Comput. **6** (1997), no. 2, 145–152. 1

[DFKO07] Irit Dinur, Ehud Friedgut, Guy Kindler, and Ryan O'Donnell, *On the Fourier tails of bounded functions over the discrete cube*, Israel J. Math. **160** (2007), 389–412. 2, 7, 8, 9, 10

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, *A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem*, 2014, arXiv:1412.6062. 2, 13

[Hås00] Johan Håstad, *On bounded occurrence constraint satisfaction*, Inform. Process. Lett. **74** (2000), no. 1-2, 1–6. 1, 2

[Hås01] ______, *Some optimal inapproximability results*, J. ACM **48** (2001), no. 4, 798–859. 1

[HV04] Johan Håstad and S. Venkatesh, *On the advantage over a random assignment*, Random Structures Algorithms **25** (2004), no. 2, 117–149. 1

[KN08] Subhash Khot and Assaf Naor, *Linear equations modulo 2 and the L_1 diameter of convex bodies*, SIAM J. Comput. **38** (2008), no. 4, 1448–1463. 1, 4, 6

[O'D14] Ryan O'Donnell, *Analysis of Boolean functions*, Cambridge University Press, 2014. 5, 6

[She92] James B. Shearer, *A note on bipartite subgraphs of triangle-free graphs*, Random Structures Algorithms **3** (1992), no. 2, 223–226. 1

ECCC ISSN 1433-8092

http://eccc.hpi-web.de